Apprentissage et calibrage d'une heuristique générique de choix de valeur dans un solveur de programmation par contraintes.

Tom Marty[1,2,3], Léo Boisvert[1,3], Tristan François[2], Pierre Tessier[2], Louis Gautier[2], Louis-Martin Rousseau[1], Quentin Cappart[1,3]

[1] Polytechnique Montréal, Montreal, Canada
[2] École Polytechnique, Palaiseau, France
[3] MILA (Institut québécois d'intelligence artificielle), Montreal, Canada

23 avril 2025

Résumé

Ce papier est un résumé de l'article "Learning and Fine-Tuning a Generic Value-Selection Heuristic Inside a Constraint Programming Solver", publié dans la revue "Constraints" en novembre 2024. Notre contribution est une méthode pour apprendre des heuristiques de sélection de valeurs dans un solveur de programmation par contraintes (CP). Pour cela, nous utilisons (1) Une procédure d'apprentissage basée sur des redémarrage avec de l'apprentissage par renforcement, (2) Une fonction de récompense capable d'attribuer des valeurs intermédiaires non nulles basées sur la propagation, et (3) une architecture neuronale basée sur un réseau de neurones à graphe hétérogène.

Mots-clés

GNNs, Apprentissage par renforcement

1 Introduction et Contexte

Dans cet article, nous présentons une méthode pour apprendre des heuristiques de sélection de valeurs dans un solveur de programmation par contraintes (CP). Cette recherche est motivée par un constat : si la programmation par contraintes est une approche efficace pour résoudre des problèmes combinatoires, son efficacité dépend fortement des heuristiques de branchement utilisées.

Les heuristiques actuelles, comme la recherche basée sur l'impact [12] ou l'activité [10], présentent des limitations : elles nécessitent une initialisation coûteuse en calcul et offrent de faibles performances au début de la recherche. De plus, le développement d'heuristiques spécifiques à un problème exige une expertise approfondie, souvent inaccessible pour les utilisateurs sans formation avancée en programmation par contraintes. Cette observation a motivé de nombreux travaux visant à utiliser l'apprentissage automatique pour concevoir automatiquement des heuristiques efficaces sans intervention experte [2].

Notre contribution principale est une procédure d'apprentissage générique qui permet d'obtenir une heuristique de sélection de valeurs directement à partir d'un modèle CP en entrée. Notre approche combine trois éléments clés :

1. Une procédure d'apprentissage basée sur des redémarrages pour entraîner un agent d'apprentissage par renforcement [15] à l'intérieur d'un solveur CP.
2. Une fonction de récompense capable d'attribuer des valeurs intermédiaires non nulles basées sur la propagation effectuée pendant la recherche.
3. Une architecture neuronale basée sur une représentation de graphe tripartite et un réseau de neurones à graphe hétérogène, inspirée des travaux sur MIPs [8].

2 Contributions Techniques

2.1 Apprentissage basé sur les redémarrages

Contrairement aux approches précédentes [5] qui définissent un épisode d'apprentissage comme un processus de résolution complet utilisant le DFS, nous proposons une approche basée sur les redémarrages. Dans notre méthode, un épisode est défini comme une seule *plongée* dans l'arbre de recherche, sans retour arrière. L'épisode s'arrête lorsqu'une solution complète est trouvée ou lorsqu'un échec résulte de la dernière décision de branchement.

Cette approche présente des avantages significatifs. Premièrement, elle évite les difficultés liées à l'utilisation d'un algorithme de retour arrière dans un cadre de processus de décision markovien, problème souligné précédemment [13]. Deuxièmement, elle permet de focaliser l'apprentissage sur l'objectif de trouver rapidement de bonnes solutions, plutôt que de prouver l'optimalité (qui nécessite souvent l'exploration d'un arbre de recherche exponentiellement grand).

S'inspirant des travaux récents [4, 13], notre méthode exécute chaque épisode à l'intérieur du solveur pendant l'entraînement, ce qui nous permet d'exploiter les informations mises à jour lors de la propagation pour le branchement.

2.2 Récompense basée sur la propagation

Nous avons développé un nouveau schéma de récompense basé sur la réduction du domaine de la variable choisie. Cette réduction se produit au moment de la propagation des contraintes après avoir sélectionné une valeur.

Notre fonction de récompense comprend deux composantes :

1. Une récompense intermédiaire recueillie aux nœuds de

branchement (mentionné plus haut).
2. Une récompense terminale obtenue en fin d'épisode.

Pour un problème de minimisation, la récompense intermédiaire suit deux principes : toute réduction de domaine des valeurs les plus élevées est récompensée et toute réduction des plus petites est pénalisée. Cette approche guide l'agent vers des situations où le coût minimum peut être éventuellement obtenu tout en éliminant les solutions coûteuses.
Ce mécanisme de récompense intermédiaire est particulièrement important car il permet d'atténuer le problème bien connu des récompenses clairsemées (*sparse rewards*), qui complique généralement l'apprentissage par renforcement.

2.3 Architecture de GNN hétérogène

Notre architecture neuronale est conçue pour effectuer une sélection de valeur. Elle comprend quatre étapes :

1. **Encodeur de modèle CP** : Nous représentons le modèle CP comme un graphe tripartite où les variables, les contraintes et les valeurs correspondent à des types de nœuds spécifiques et des arêtes représentant leurs relations. Cette approche s'inspire des travaux sur les MIPs [8], adaptée à la programmation par contraintes, qui implique des variables non binaires et des contraintes combinatoires. Notre encodage se distingue d'approches récentes [3, 16], offrant un meilleur équilibre entre expressivité et efficacité computationnelle.
2. **Encodeur de réseau neuronal à graphe** : Nous utilisons une architecture hétérogène où une convolution est effectuée pour chaque type de nœud. Ceci permet de propager l'information dans le graphe tout en préservant les caractéristiques de chaque type de nœud.
3. **Décodeur de réseau neuronal** : Cette étape prédit une valeur Q pour chaque valeur disponible pour la variable sélectionnée, en utilisant un algorithme d'apprentissage par renforcement profond (*deep Q-learning*) [11].
4. **Politique d'action** : Cette politique définit comment exploiter les valeurs Q approximées, équilibrant exploration et exploitation pendant la phase d'apprentissage.

3 Résultats Expérimentaux

Nous avons évalué notre approche sur quatre problèmes d'optimisation combinatoire : coloration de graphes, ensemble indépendant maximum (MIS), coupe maximum (MAXCUT), couverture de sommets minimum (MVC).
Les expériences ont été menées sur des instances de différentes tailles : petites (20-30 nœuds), moyennes (40-50 nœuds) et grandes (80-100 nœuds). L'entraînement a été effectué sur des graphes générés aléatoirement selon le modèle de Barabási-Albert [1], connu pour générer des graphes qui imitent les organisations humaines et naturelles. Nos résultats montrent que notre heuristique apprise :

— Trouve des solutions de qualité supérieure étant donné un budget de nœuds.
— Trouve la solution optimale en explorant moins de nœuds que les méthodes de référence [10, 12].
— Apprend une stratégie de branchement donnant des solutions de haute qualité, après une seule plongée.

Pour l'évaluation, nous avons utilisé des profils de performance [7] qui montrent que notre méthode surpasse les autres approches pour trouver rapidement de bonnes solutions. Par exemple, pour le problème de MAXCUT avec 50 nœuds, une seule plongée avec notre approche produit une solution avec un écart d'optimalité de 0.16, alors qu'une recherche en profondeur avec une sélection de valeur aléatoire nécessite 19 secondes et 53 000 nœuds explorés pour trouver une solution avec un écart similaire.

Les expériences montrent également que l'intégration de notre heuristique apprise dans une recherche de type ILDS [9] améliore considérablement les résultats, permettant d'obtenir des solutions très proches de l'optimalité. Nous observons finalement que le calibrage (*fine-tuning*) d'un modèle avec une classe de problèmes différente peut accélérer considérablement le processus d'apprentissage [14], comme démontré sur le problème de coupe maximum avec 80 nœuds.

4 Défis et Perspectives

Malgré les résultats prometteurs, plusieurs défis persistent.

1. **Mise à l'échelle de la représentation** : La taille des graphes augmente significativement avec la taille du problème et le nombre de nœuds dans l'arbre de recherche croît exponentiellement. Des mécanismes de compression pourraient aider à atténuer cette difficulté.
2. **Problèmes fortement contraints** : Notre approche excelle avec des solutions facilement réalisables, mais perd en efficacité sur des problèmes fortement contraints où les récompenses sont plus rares.
3. **Heuristique combinée variable/valeur** : Une direction de recherche intéressante serait d'adapter notre architecture pour apprendre simultanément une heuristique de sélection de variables et de valeurs.
4. **Preuve d'optimalité** : Notre objectif actuel est de trouver rapidement une bonne solution, mais accélérer la preuve d'optimalité reste un défi ouvert. La redéfinition de la fonction de récompense ou de l'épisode pourrait aider à aborder ce problème.
5. **Architecture de GNN** : Les GNNs basés sur le passage de messages ont des limitations intrinsèques concernant leur capacité de représentation [6] . L'exploration d'architectures alternatives est une voie prometteuse.

5 Conclusion

Nous avons proposé une méthode générique basée sur l'apprentissage par renforcement pour obtenir une heuristique de sélection de valeurs. Notre approche combine un entraînement basé sur les redémarrages, un signal de récompense non clairsemé, et une architecture de GNN hétérogène.

Les expériences sur quatre problèmes démontrent que notre méthode peut trouver de meilleures solutions en visitant significativement moins de nœuds que d'autres méthodes de référence. Nous avons également identifié plusieurs défis qui constituent des directions pour nos travaux futurs.

Références

[1] Réka Albert and Albert-László Barabási. Statistical mechanics of complex networks. *Reviews of Modern Physics*, 74(1) :47–97, 2002.

[2] Yoshua Bengio, Andrea Lodi, and Antoine Prouvost. Machine learning for combinatorial optimization : a methodological tour d'horizon. *European Journal of Operational Research*, 290(2) :405–421, 2021.

[3] Léo Boisvert, Hélène Verhaeghe, and Quentin Cappart. Towards a generic representation of combinatorial problems for learning-based approaches. In *International Conference on the Integration of Constraint Programming, Artificial Intelligence, and Operations Research*, pages 99–108. Springer, 2024.

[4] Quentin Cappart, Thierry Moisan, Louis-Martin Rousseau, Isabeau Prémont-Schwarz, and Andre A. Cire. Combining reinforcement learning and constraint programming for combinatorial optimization. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 3677–3687, 2021.

[5] Félix Chalumeau, Ilan Coulon, Quentin Cappart, and Louis-Martin Rousseau. Seapearl : A constraint programming solver guided by reinforcement learning. In *International Conference on Integration of Constraint Programming, Artificial Intelligence, and Operations Research*, pages 392–409. Springer, 2021.

[6] Ziang Chen, Jialin Liu, Xiaohan Chen, Xinshang Wang, and Wotao Yin. Rethinking the capacity of graph neural networks for branching strategy. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024.

[7] Elizabeth D Dolan and Jorge J Moré. Benchmarking optimization software with performance profiles. *Mathematical programming*, 91 :201–213, 2002.

[8] Maxime Gasse, Didier Chetelat, Nicola Ferroni, Laurent Charlin, and Andrea Lodi. Exact combinatorial optimization with graph convolutional neural networks. In *Advances in Neural Information Processing Systems*, pages 15580–15592, 2019.

[9] William D Harvey and Matthew L Ginsberg. Limited discrepancy search. In *IJCAI*, volume 95, pages 607–615, 1995.

[10] Laurent Michel and Pascal Van Hentenryck. Activity-based search for black-box constraint programming solvers. *Integration of AI and OR Techniques in Contraint Programming for Combinatorial Optimzation Problems*, pages 228–243, 2012.

[11] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *nature*, 518(7540) :529–533, 2015.

[12] Philippe Refalo. Impact-based search strategies for constraint programming. In *International Conference on Principles and Practice of Constraint Programming*, pages 557–571. Springer, 2004.

[13] Wen Song, Zhiguang Cao, Jie Zhang, Chi Xu, and Andrew Lim. Learning variable ordering heuristics for solving constraint satisfaction problems. *Engineering Applications of Artificial Intelligence*, 109 :104603, 2022.

[14] Yifei Sun, Qi Zhu, Yang Yang, Chunping Wang, Tianyu Fan, Jiajun Zhu, and Lei Chen. Fine-tuning graph neural networks by preserving graph generative patterns. *Proceedings of the AAAI Conference on Artificial Intelligence*, 38(8) :9053–9061, 2024.

[15] Richard S Sutton and Andrew G Barto. Reinforcement learning : an introduction. a bradford book. *Cambridge, MA, USA*, 262039249, 2018.

[16] Jan Tönshoff, Berke Kisin, Jakob Lindner, and Martin Grohe. One model, any csp : graph neural networks as fast global search heuristics for constraint satisfaction. In *Proceedings of the Thirty-Second International Joint Conference on Artificial Intelligence*, IJCAI '23, 2023.