

Apprentissage de bornes duales valides en programmation par contrainte : Décomposition lagrangienne amplifiée avec apprentissage auto-supervisé

Swann Bessa^{1,2}, Darius Dabert^{1,2}, Max Bourgeat¹, Louis-Martin Rousseau¹, Quentin Cappart¹

¹ Polytechnique Montréal, Montréal, Canada

² École Polytechnique, Palaiseau, France

12 mai 2025

Résumé

Ce papier est un résumé de l'article "Learning Valid Dual Bounds in Constraint Programming : Boosted Lagrangian Decomposition with Self-Supervised Learning" publié à AAAI 2025. La décomposition lagrangienne relaxe les problèmes d'optimisation en sous-problèmes plus simples pour améliorer les algorithmes de séparation et d'évaluation en permettant le calcul d'une borne duale. Toutefois, en programmation par contraintes, l'optimisation des multiplicateurs de Lagrange est coûteuse en raison de la complexité de résolution des sous-problèmes. Nous proposons une approche d'apprentissage auto-supervisé utilisant des réseaux de neurones pour générer ces multiplicateurs et obtenir des bornes plus serrées. Cela réduit le nombre d'itérations nécessaires et accélère les solveurs. Notre méthode démontre une bonne généralisation sur des problèmes comme le sac à dos multidimensionnel.

1 Introduction

L'optimisation combinatoire est utilisée dans divers domaines comme l'aérospatial, la planification des transports, l'ordonnancement et l'économie. Un défi majeur est l'explosion combinatoire : le nombre de solutions possibles croît exponentiellement avec la taille du problème, rendant la résolution de problèmes à grande échelle difficile. La programmation par contraintes (CP) est un outil versatile qui permet de traiter des problèmes combinatoires complexes, y compris ceux avec des contraintes non linéaires, grâce à des techniques de recherche et de propagation [7]. Cependant, contrairement à la programmation en nombres entiers, la programmation par contraintes ne dispose pas d'un mécanisme efficace pour obtenir des bornes duales, limitant ainsi sa compétitivité sur de nombreux problèmes [2, 1, 8, 3]. Cet article propose une approche pour combler cette lacune en développant une méthode générale et efficace de bornage dual pour la programmation par contraintes.

2 Décomposition lagrangienne en CP

Un problème d'optimisation sous contraintes (COP) est défini par un ensemble de variables discrètes X , leurs do-

maines de valeurs $D(X)$, un ensemble de contraintes C limitant les affectations possibles, et une fonction objectif $f(X)$ à optimiser. Une solution faisable est une affectation des valeurs de $D(X)$ à X respectant toutes les contraintes. Une solution optimale est une solution faisable qui maximise (ou minimise) la fonction objectif. La formulation mathématique d'un COP inclut généralement m contraintes qui définissent l'espace des solutions admissibles.

$$\max_{X \in D(X)} \left\{ f(X) \mid \bigwedge_{i=1}^m C_i(X) \right\} \quad (1)$$

La notation $C_i(X)$ indique que la contrainte C_i s'applique aux variables X . La décomposition lagrangienne (LD) consiste à relaxer le problème afin d'obtenir une borne supérieure valide et, idéalement, serrée. Pour ce faire, les variables impliquées dans chaque contrainte sont dupliquées, à l'exception de celles de la première contrainte. On obtient alors la borne duale suivante :

$$\mathcal{B}(\mu) = \max \left\{ f(X_1) + \sum_{i=2}^m \mu_i \cdot (X_1 - X_i) \mid \bigwedge_{i=1}^m C_i(X_i) \right\} \quad (2)$$

où les μ_i sont les multiplicateurs de Lagrange. Le problème résultant est une relaxation du problème initial, fournissant une borne duale $\mathcal{B}(\mu)$, qui correspond à une borne supérieure dans un problème de maximisation. Chaque contrainte C_i ayant un ensemble de variables distinct, les sous-problèmes peuvent être résolus indépendamment, simplifiant ainsi la résolution. Ainsi, Une borne duale valide est obtenue en résolvant m sous-problèmes plus simples, mais la qualité de cette borne dépend fortement des multiplicateurs de Lagrange utilisés. Trouver ces multiplicateurs optimaux est le principal défi de la décomposition lagrangienne, car l'équation associée n'est pas différentiable. La méthode la plus courante pour les ajuster est l'optimisation par sous-gradient, qui les met à jour de manière itérative à partir d'une valeur initiale arbitraire [9].

$$\mu_i^{t+1} = \mu_i^t + \alpha(X_1^t - X_i^t) \quad \forall i \in \{2, \dots, m\} \quad (3)$$

Hà et al. (2015) [4] ont proposé d'intégrer cette procédure dans la phase de recherche d'un solveur de programmation par contraintes (CP), en appliquant la décomposition lagrangienne (LD) et l'optimisation par sous-gradient à chaque nœud de l'arbre de recherche. Cependant, cette approche est très coûteuse en temps, car chaque itération du sous-gradient nécessite de résoudre chaque sous-problème, qui peuvent être NP-difficile. Nous proposons d'accélérer considérablement ce processus en utilisant de l'apprentissage automatique.

3 Notre Approche

La borne calculée dans l'Équation (2) possède deux propriétés importantes : (1) elle peut être paramétrée à l'aide des multiplicateurs de Lagrange μ et (2) elle est toujours valide, c'est-à-dire qu'elle ne sous-estimera jamais le profit réel. Ces propriétés ouvrent la voie à une approche basée sur l'apprentissage pour calculer cette borne. Inspirés par Parjadis et al. (2024) [6], nous proposons d'entraîner un modèle $\Omega_\theta : G(V, E) \rightarrow R^{|V|}$ capable de prédire directement tous les multiplicateurs μ pour un problème d'optimisation combinatoire (COP) donné sous forme de graphe G . Ce graphe est constitué d'un ensemble de nœuds V (un par multiplicateur) et d'arêtes E reliant les nœuds partageant la même variable ou la même contrainte. Le modèle Ω_θ est calibré par un ensemble de paramètres différentiables θ . L'objectif est d'éliminer les itérations du sous-gradient en apprenant directement les multiplicateurs qui produisent une borne serrée, ce qui permet d'accélérer l'exécution et d'améliorer le filtrage. La différentiabilité du modèle est essentielle, car la borne est calculée via une optimisation basée sur le gradient. L'objectif est de trouver des paramètres θ minimisant la borne. Grâce à la deuxième propriété, la borne obtenue a la garantie d'être valide, indépendamment de la précision du modèle. Ce travail introduit ainsi une méthode générique d'apprentissage de bornes duales valides pour tout type de COP discret, ce qui est un atout majeur étant donné les difficultés à obtenir des garanties en optimisation combinatoire avec l'apprentissage automatique [5]. La minimisation de la borne en fonction des multiplicateurs μ et son gradient sont ainsi formulés :

$$\min_{\theta} \mathcal{B}(\mu) \mapsto \nabla_{\theta} \mathcal{B}(\mu) \text{ avec } \mu = \Omega_{\theta}(G) \quad (4)$$

En appliquant la règle du chaînage on obtient alors :

$$\nabla_{\theta} \mathcal{B}(\mu) = \left\langle (X_1 - X_i) \cdot \frac{\partial \mu_i}{\partial \theta}, \dots \right\rangle \forall i \in \{2, \dots, m\} \quad (5)$$

Nous proposons d'utiliser une approche d'apprentissage auto-supervisé pour paramétrer le modèle Ω_θ . La procédure d'entraînement est formalisée comme suit. En entrée, elle reçoit un ensemble de données D composé de graphes $G(V, E)$ représentant des instances d'un problème d'optimisation combinatoire (COP). Ces instances peuvent provenir de données historiques ou être générées synthétiquement, et leurs caractéristiques dépendent du problème

traité. À chaque itération, une étape d'optimisation L_D est réalisée, et tous les sous-problèmes sont résolus via une procédure dédiée afin d'obtenir les valeurs optimales de X_i pour les multiplicateurs μ actuels. Ensuite, la borne et son gradient sont calculés, suivis d'une mise à jour par descente de gradient. Cela ajuste les valeurs des multiplicateurs pour l'itération suivante, modifiant ainsi les solutions optimales des variables X_i . Finalement, les paramètres θ du réseau de neurones entraîné Ω_θ sont retournés.

4 Expérimentation

La méthode est testée sur deux problèmes d'optimisation combinatoire difficiles : le problème du sac à dos multidimensionnel et le problème de planification des horaires. Nous testons différentes méthodes :

- **CP** : Une approche pure de programmation par contraintes sans décomposition lagrangienne ni apprentissage.
- **CP+SG** : Le même modèle CP, amélioré avec la décomposition lagrangienne et l'optimisation par sous-gradient, comme proposé par Hà et al. (2015) [4].
- **CP+Learning(all)** : Le modèle CP avec décomposition lagrangienne, utilisant les bornes apprises au lieu du sous-gradient, avec l'apprentissage appliqué à chaque nœud de l'arbre de recherche.
- **CP+Learning(all)+SG** : Similaire à l'approche précédente, mais les bornes apprises sont encore améliorées par l'optimisation sous-gradient. Le modèle entraîné Ω_θ est appelé à chaque nœud de l'arbre de recherche pour obtenir les bornes.
- **CP+Learning(root)+SG** : Le modèle Ω_θ est utilisé uniquement au nœud racine, la borne résultante servant de valeur initiale pour amorcer l'optimisation sous-gradient dans les autres nœuds.

Les résultats obtenus confirment que l'on obtient en moyenne des bornes de meilleures qualités et que notre modèle a une bonne capacité à s'adapter à des instances dont la distribution n'a pas été rencontrée durant la phase d'entraînement.

5 Conclusion

L'article propose une méthode innovante pour améliorer l'efficacité de la décomposition lagrangienne dans la programmation par contraintes. Il utilise un réseau de neurones pour apprendre automatiquement des multiplicateurs lagrangiens, permettant de produire des bornes duales serrées sans nécessiter de bornes étiquetées. Cette approche auto-supervisée permet soit de remplacer entièrement les itérations de sous-gradient, soit de les amorcer afin d'accélérer la convergence. Les expériences montrent que la méthode réduit le temps d'exécution tout en conservant des bornes de qualité. De plus, un ajustement fin permet une meilleure généralisation à des instances hors distribution. Ces résultats ouvrent des perspectives prometteuses pour appliquer cette technique à des problèmes combinatoires où la programmation par contraintes est performante.

Références

- [1] Pascal Benchimol, Willem-Jan van Hoeve, Jean-Charles Régin, Louis-Martin Rousseau, and Michel Rueher. Improved filtering for weighted circuit constraints. *Constraints*, 17 :205–233, 2012.
- [2] Filippo Focacci, Andrea Lodi, and Michela Milano. Cost-based domain filtering. *Principles and Practice of Constraint Programming–CP’99 : 5th International Conference, CP’99, Alexandria, VA, USA, October 11–14, 1999. Proceedings 5*, 1999.
- [3] Monique Guignard and Siwhan Kim. Lagrangean decomposition : A model yielding stronger lagrangean bounds. *Mathematical programming*, 39(2) :215–228, 1987.
- [4] Minh Hoàng Hà, Claude-Guy Quimper, and Louis-Martin Rousseau. General bounding mechanism for constraint programs. In *Principles and Practice of Constraint Programming : 21st International Conference, CP 2015, Cork, Ireland, August 31–September 4, 2015, Proceedings 21*, pages 158–172. Springer, 2015.
- [5] James Kotary, Ferdinando Fioretto, Pascal van Hentenryck, and Bryan Wilder. End-to-end constrained optimization learning : A survey. In *30th International Joint Conference on Artificial Intelligence, IJCAI 2021*, pages 4475–4482. International Joint Conferences on Artificial Intelligence, 2021.
- [6] Augustin Parjadis, Quentin Cappart, Bistra Dilkina, Aaron Ferber, and Louis-Martin Rousseau. Learning Lagrangian Multipliers for the Travelling Salesman Problem. In Paul Shaw, editor, *30th International Conference on Principles and Practice of Constraint Programming (CP 2024)*, volume 307 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 22 :1–22 :18, Dagstuhl, Germany, 2024. Schloss Dagstuhl – Leibniz-Zentrum für Informatik.
- [7] Francesca Rossi, Peter Van Beek, and Toby Walsh. *Handbook of constraint programming*. Elsevier, 2006.
- [8] Meinolf Sellmann and Torsten Fahle. CP-based lagrangian relaxation for a multimedia application. In *Third International Workshop on the Integration of AI and OR Techniques (CPAIOR 2001)*, 2001.
- [9] Naum Zuselevich Shor. *Minimization methods for non-differentiable functions*, volume 3. Springer Science & Business Media, 2012.