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

Swann Bessa, Darius Dabert, Max Bourgeat, Louis-Martin Rousseau, Quentin Cappart

JFPC 2025

POLYTECHNIQUE MONTRÉAL

UNIVERSITÉ D'INGÉNIERIE

Programmation par Contraintes (CP)

Un problème de programmation par contraintes c'est :

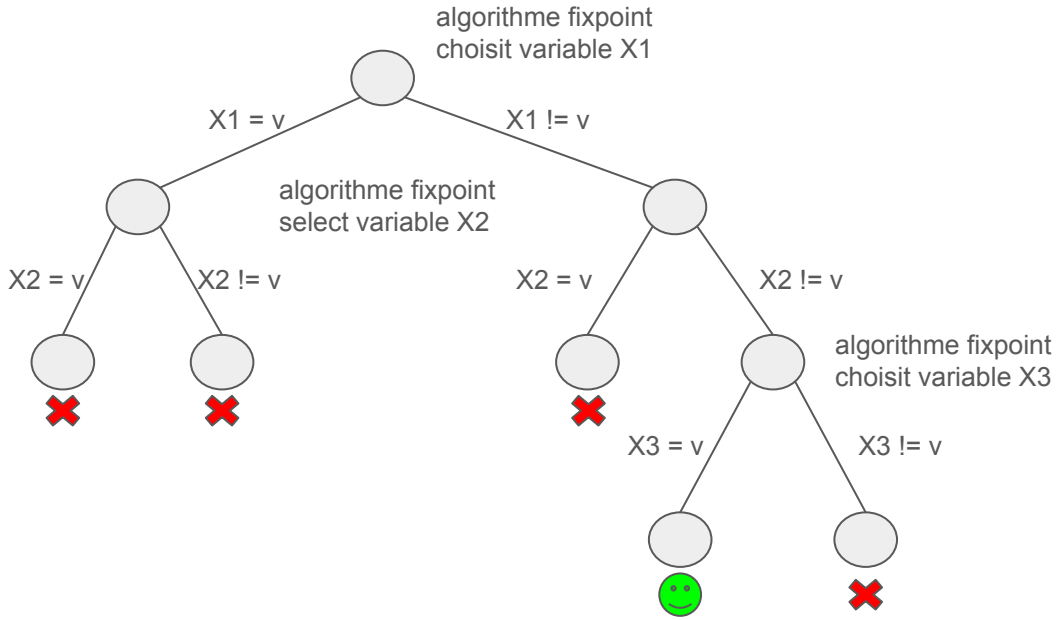
1) Variables
2) Domaines des variables
3) Contraintes

$$
\begin{aligned}
\max \quad & f(X_1, X_2, X_3) \\
\text{s.à} \quad & C_1(X_1, X_2, X_3) \\
& C_2(X_2, X_3) \\
& X_1, X_2, X_3 \in \mathbb{N}
\end{aligned}
$$

Un problème de programmation par contraintes c'est :

1) Variables
2) Domaines des variables
3) Contraintes

$$\begin{aligned} \max \quad & f(X_1, X_2, X_3) \\ \text{s.à} \quad & C_1(X_1, X_2, X_3) \\ & C_2(X_2, X_3) \\ & X_1, X_2, X_3 \in \mathbb{N} \end{aligned}$$



L'algorithme du fixpoint propage progressivement les contraintes jusqu'à ne plus pouvoir réduire les domaines

Permet de résoudre des Problèmes Combinatoires !

Décomposition Lagrangienne (LD)

La décomposition lagrangienne décompose le problèmes en sous problèmes indépendants et plus simples

$$
\begin{aligned}
\max \quad & f(X_1, X_2, X_3) \\
\text{s.à} \quad & C_1(X_1, X_2, X_3) \\
& C_2(X_2, X_3) \\
& X_1, X_2, X_3 \in \mathbb{N}
\end{aligned}
$$

La décomposition lagrangienne décompose le problèmes en sous problèmes indépendants et plus simples

Étape 1 : chaque variable de chaque contrainte est dupliquée, sauf pour la première contrainte

$$\begin{aligned} \max \quad & f(X_1, X_2, X_3) \\ \text{s.à} \quad & C_1(X_1, X_2, X_3) \\ & C_2(X_2, X_3) \\ & X_1, X_2, X_3 \in \mathbb{N} \end{aligned}$$

$$\begin{aligned} \max \quad & f(X_1, X_2, X_3) \\ \text{s.à} \quad & C_1(X_1, X_2, X_3) \\ & C_2(Y_2, Y_3) \\ & X_1, X_2, X_3, Y_2, Y_3 \in \mathbb{N} \end{aligned}$$

La décomposition lagrangienne décompose le problèmes en sous problèmes indépendants et plus simples

Étape 1 : chaque variable de chaque contrainte est dupliquée, sauf pour la première contrainte

Étape 2 : une contrainte liant les valeurs est ajoutée pour chaque nouvelle variable

$$\begin{aligned} \max \quad & f(X_1, X_2, X_3) \\ \text{s.à} \quad & C_1(X_1, X_2, X_3) \\ & C_2(X_2, X_3) \\ & X_1, X_2, X_3 \in \mathbb{N} \end{aligned}$$

$$\begin{aligned} \max \quad & f(X_1, X_2, X_3) \\ \text{s.à} \quad & C_1(X_1, X_2, X_3) \\ & C_2(Y_2, Y_3) \\ & X_2 = Y_2, X_3 = Y_3 \\ & X_1, X_2, X_3, Y_2, Y_3 \in \mathbb{N} \end{aligned}$$

La décomposition lagrangienne décompose le problèmes en sous problèmes indépendants et plus simples

Étape 1 : chaque variable de chaque contrainte est dupliquée, sauf pour la première contrainte

Étape 2 : une contrainte liant les valeurs est ajoutée pour chaque nouvelle variable

Étape 3: ces contraintes sont déplacées dans la fonction objectif avec une pénalité

$$\begin{aligned} \max \quad & f(X_1, X_2, X_3) \\ \text{s.à} \quad & C_1(X_1, X_2, X_3) \\ & C_2(X_2, X_3) \\ & X_1, X_2, X_3 \in \mathbb{N} \end{aligned}$$

$$\begin{aligned} \max \quad & f(X_1, X_2, X_3) \\ \text{s.à} \quad & C_1(X_1, X_2, X_3) \\ & C_2(Y_2, Y_3) \\ & X_2 = Y_2, X_3 = Y_3 \\ & X_1, X_2, X_3, Y_2, Y_3 \in \mathbb{N} \end{aligned}$$

$$f(X_1, X_2, X_3) + \mu_2 (Y_2 - X_2) + \mu_3 (Y_3 - X_3)$$

La décomposition lagrangienne décompose le problèmes en sous problèmes indépendants et plus simples

Étape 1 : chaque variable de chaque contrainte est dupliquée, sauf pour la première contrainte

Étape 2 : une contrainte liant les valeurs est ajoutée pour chaque nouvelle variable

Étape 3: ces contraintes sont déplacées dans la fonction objectif avec une pénalité

$$\begin{aligned} \max \quad & f(X_1, X_2, X_3) \\ \text{s.à} \quad & C_1(X_1, X_2, X_3) \\ & C_2(X_2, X_3) \\ & X_1, X_2, X_3 \in \mathbb{N} \end{aligned}$$

$$\begin{aligned} \max \quad & f(X_1, X_2, X_3) \\ \text{s.à} \quad & C_1(X_1, X_2, X_3) \\ & C_2(Y_2, Y_3) \\ & X_2 = Y_2, \; X_3 = Y_3 \\ & X_1, X_2, X_3, Y_2, Y_3 \in \mathbb{N} \end{aligned}$$

$$f(X_1, X_2, X_3) + \mu_2 (Y_2 - X_2) + \mu_3 (Y_3 - X_3)$$

La décomposition lagrangienne décompose le problèmes en sous problèmes indépendants et plus simples

Étape 1 : chaque variable de chaque contrainte est dupliquée, sauf pour la première contrainte

Étape 2 : une contrainte liant les valeurs est ajoutée pour chaque nouvelle variable

Étape 3: ces contraintes sont déplacées dans la fonction objectif avec une pénalité

$$\begin{aligned} \max \quad & f(X_1, X_2, X_3) \\ \text{s.à} \quad & C_1(X_1, X_2, X_3) \\ & C_2(X_2, X_3) \\ & X_1, X_2, X_3 \in \mathbb{N} \end{aligned}$$

$$\begin{aligned} \max \quad & f(X_1, X_2, X_3) \\ \text{s.à} \quad & C_1(X_1, X_2, X_3) \\ & C_2(Y_2, Y_3) \\ & X_2 = Y_2, X_3 = Y_3 \\ & X_1, X_2, X_3, Y_2, Y_3 \in \mathbb{N} \end{aligned}$$

$$f(X_1, X_2, X_3) + \mu_2 (Y_2 - X_2) + \mu_3 (Y_3 - X_3)$$

$$\mathcal{B}(\mu_2, \mu_3) = \begin{cases} \max & f(X_1, X_2, X_3) + \mu_2(Y_2 - X_2) + \mu_3(Y_3 - X_3) \\ \text{s.à} & C_1(X_1, X_2, X_3) \\ & C_2(Y_2, Y_3) \\ & X_1, X_2, X_3, Y_2, Y_3 \in \mathbb{N} \end{cases}$$

Résoudre ce problème relaxé fournit une **borne duale**

$$\mathcal{B}(\mu_2, \mu_3) = \begin{cases} \max & f(X_1, X_2, X_3) + \mu_2(Y_2 - X_2) + \mu_3(Y_3 - X_3) \\ \text{s.à} & C_1(X_1, X_2, X_3) \\ & C_2(Y_2, Y_3) \\ & X_1, X_2, X_3, Y_2, Y_3 \in \mathbb{N} \end{cases}$$

Résoudre ce problème relaxé fournit une **borne duale**

Conséquence : chaque contrainte peut être résolue indépendamment

$$\mathcal{B}(\mu_2, \mu_3) = \begin{cases} \max & f(X_1, X_2, X_3) - \mu_2 X_2 - \mu_3 X_3 \\ \text{s.à} & C_1(X_1, X_2, X_3) \\ & X_1, X_2, X_3 \in \mathbb{N} \end{cases} + \begin{cases} \max & \mu_2 Y_2 + \mu_3 Y_3 \\ \text{s.à} & C_2(Y_2, Y_3) \\ & Y_2, Y_3 \in \mathbb{N} \end{cases}$$

Pour des multiplicateurs donnés, on obtient une borne en résolvant ces sous problèmes

$$\mathcal{B}(\mu_2, \mu_3) = \begin{cases} \max & f(X_1, X_2, X_3) + \mu_2(Y_2 - X_2) + \mu_3(Y_3 - X_3) \\ \text{s.à} & C_1(X_1, X_2, X_3) \\ & C_2(Y_2, Y_3) \\ & X_1, X_2, X_3, Y_2, Y_3 \in \mathbb{N} \end{cases}$$

Résoudre ce problème relaxé fournit une **borne duale**

Conséquence : chaque contrainte peut être résolue indépendamment

$$\mathcal{B}(\mu_2, \mu_3) = \begin{cases} \max & f(X_1, X_2, X_3) - \mu_2 X_2 - \mu_3 X_3 \\ \text{s.à} & C_1(X_1, X_2, X_3) \\ & X_1, X_2, X_3 \in \mathbb{N} \end{cases} + \begin{cases} \max & \mu_2 Y_2 + \mu_3 Y_3 \\ \text{s.à} & C_2(Y_2, Y_3) \\ & Y_2, Y_3 \in \mathbb{N} \end{cases}$$

Pour des multiplicateurs donnés, on obtient une borne en résolvant ces sous problèmes

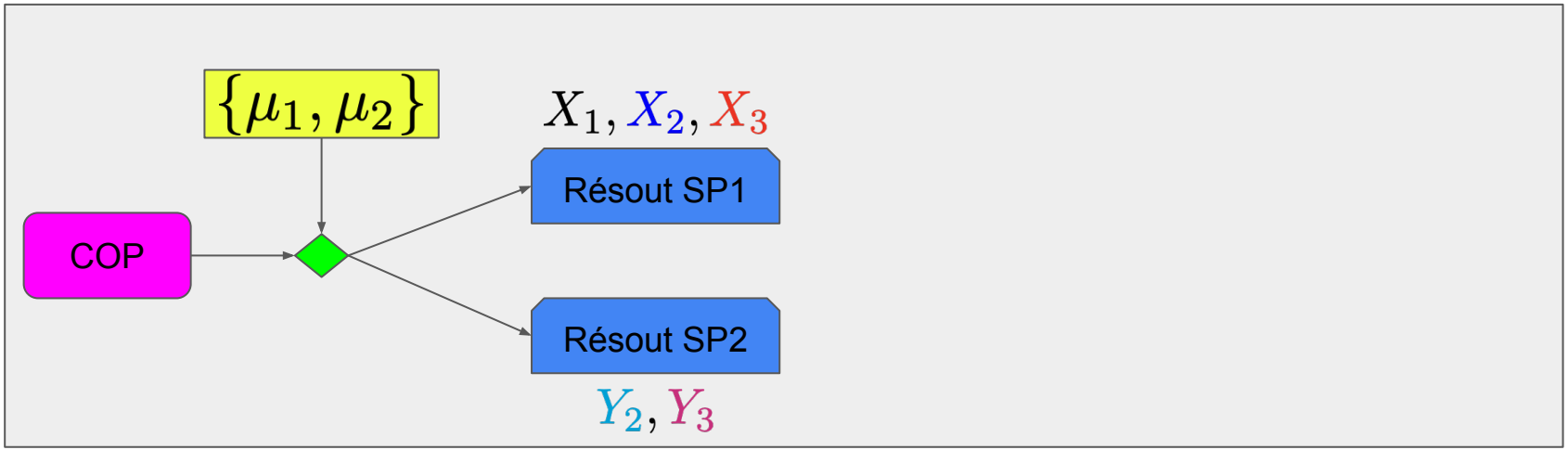
Quelles valeurs pour ces multiplicateurs ?

Décomposition Lagrangienne en CP

Initialisation : on initialise les multiplicateurs à des valeurs arbitraires

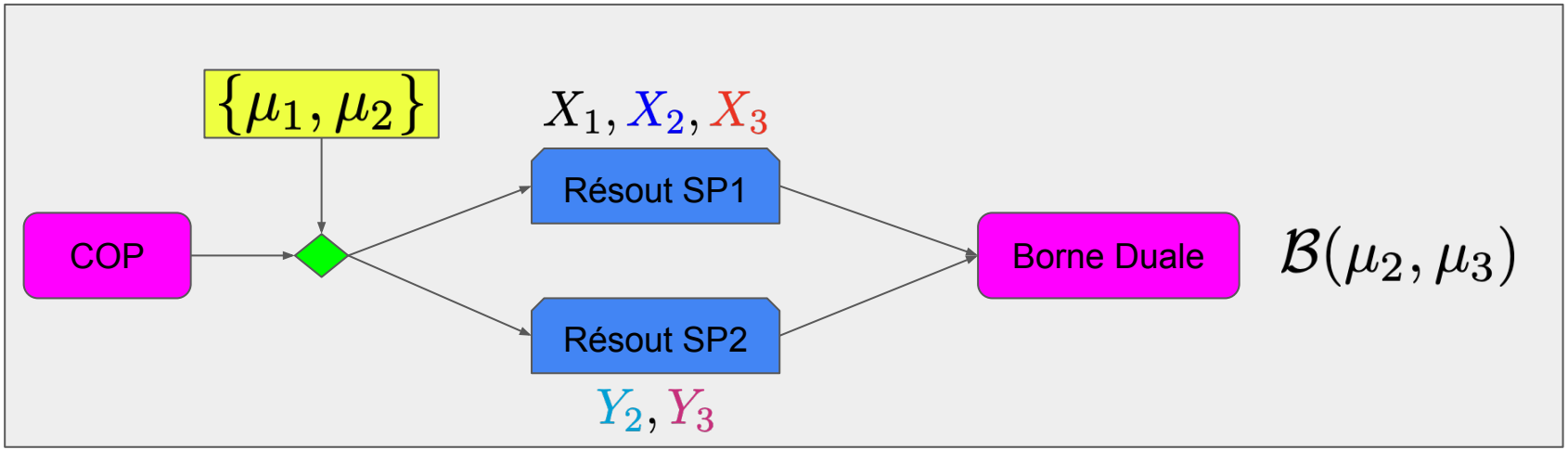
Initialisation : on initialise les multiplicateurs à des valeurs arbitraires

Étape 1 : on résout tous les sous problèmes avec ces valeurs et on obtient une borne duale



Initialisation : on initialise les multiplicateurs à des valeurs arbitraires

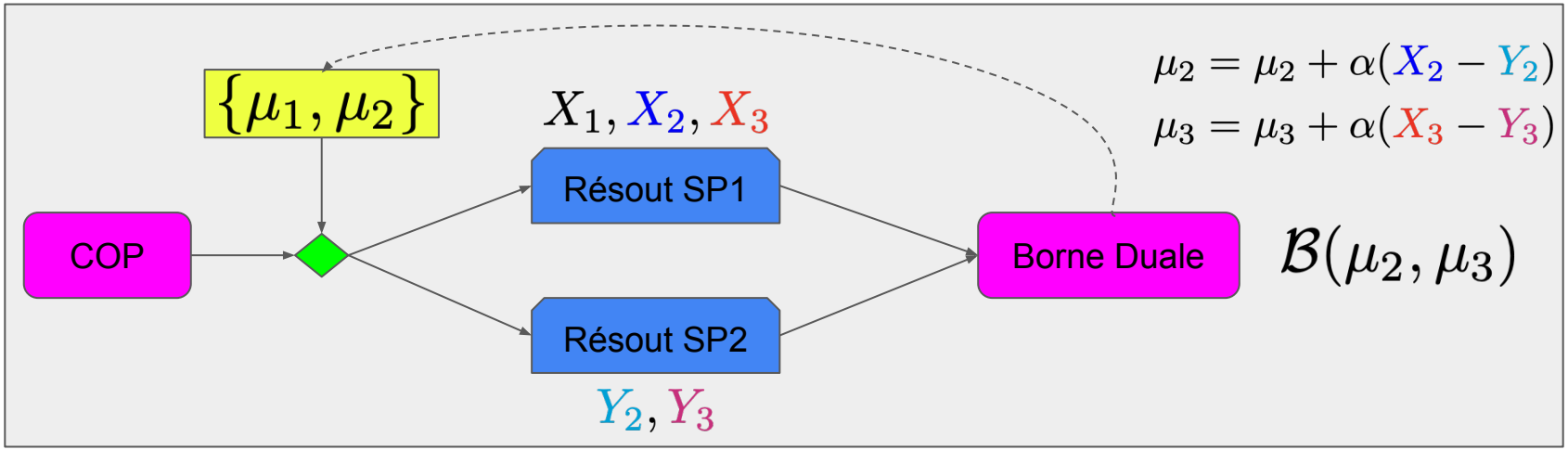
Étape 1 : on résout tous les sous problèmes avec ces valeurs et on obtient une borne duale



Initialisation : on initialise les multiplicateurs à des valeurs arbitraires

Étape 1 : on résout tous les sous problèmes avec ces valeurs et on obtient une borne duale

Étape 2 : on met à jours les multiplicateurs avec leur sous gradient

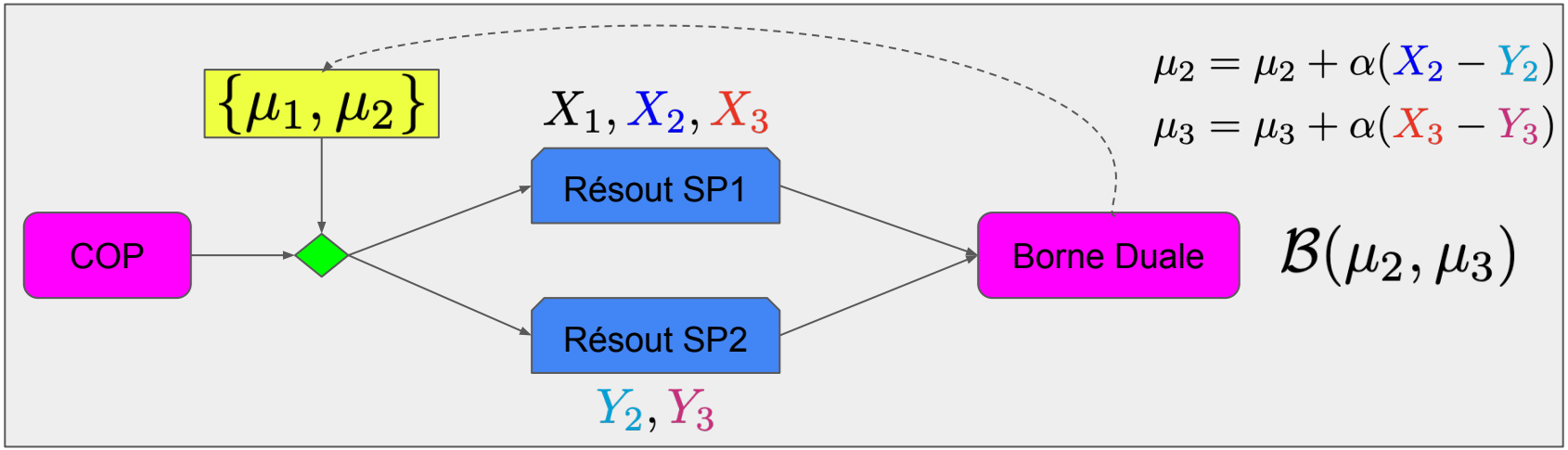


Initialisation : on initialise les multiplicateurs à des valeurs arbitraires

Étape 1 : on résout tous les sous problèmes avec ces valeurs et on obtient une borne duale

Étape 2 : on met à jours les multiplicateurs avec leur sous gradient

Étape 3 : on répète les étapes 1 et 2 pour x itérations

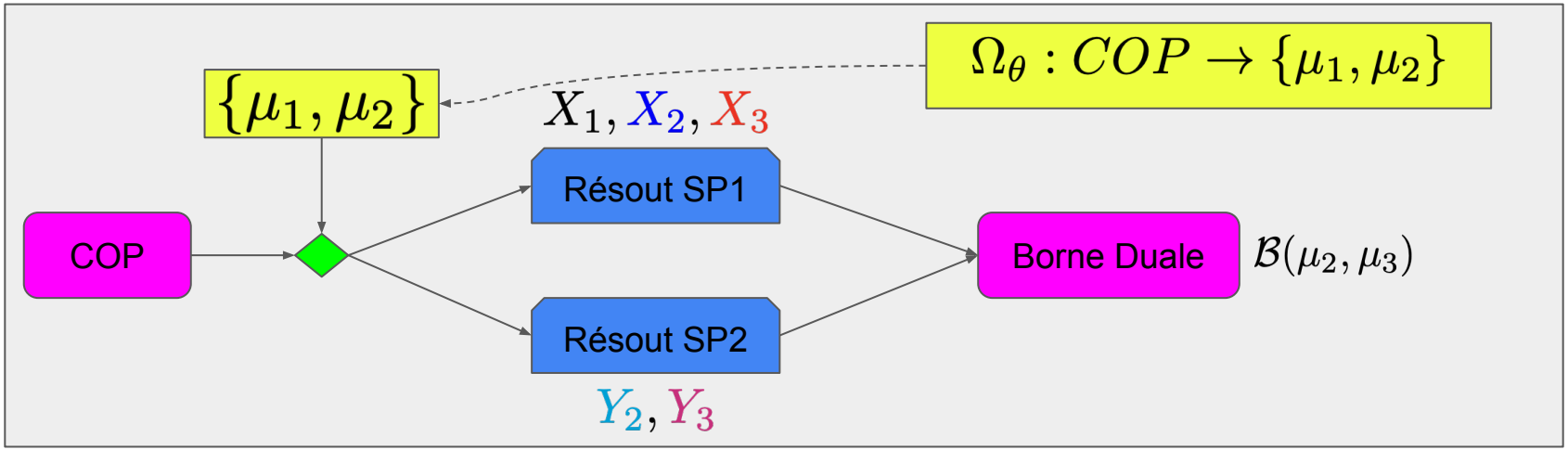


Ce processus est coûteux car il demande de résoudre beaucoup de sous-problème à chaque itération

Notre Approche

On propose une approche auto-supervisée pour calculer les multiplicateurs

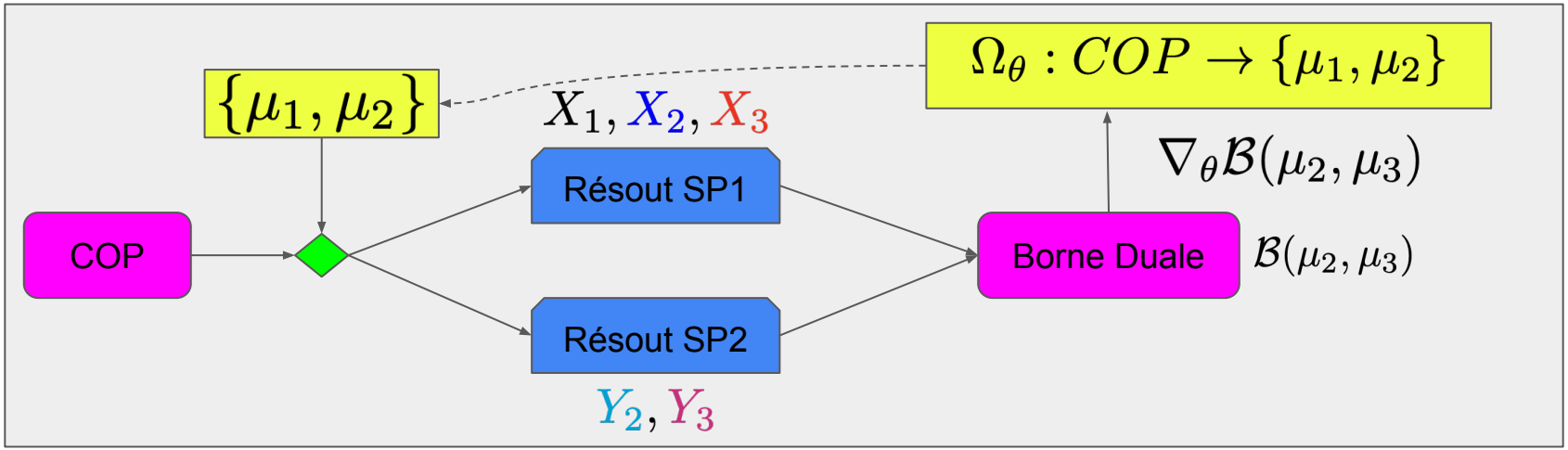
Étape 1 : les multiplicateurs sont maintenant obtenus avec un modèle prédictif différentiable



On propose une approche auto-supervisée pour calculer les multiplicateurs

Étape 1 : les multiplicateurs sont maintenant obtenus avec un modèle prédictif différentiable

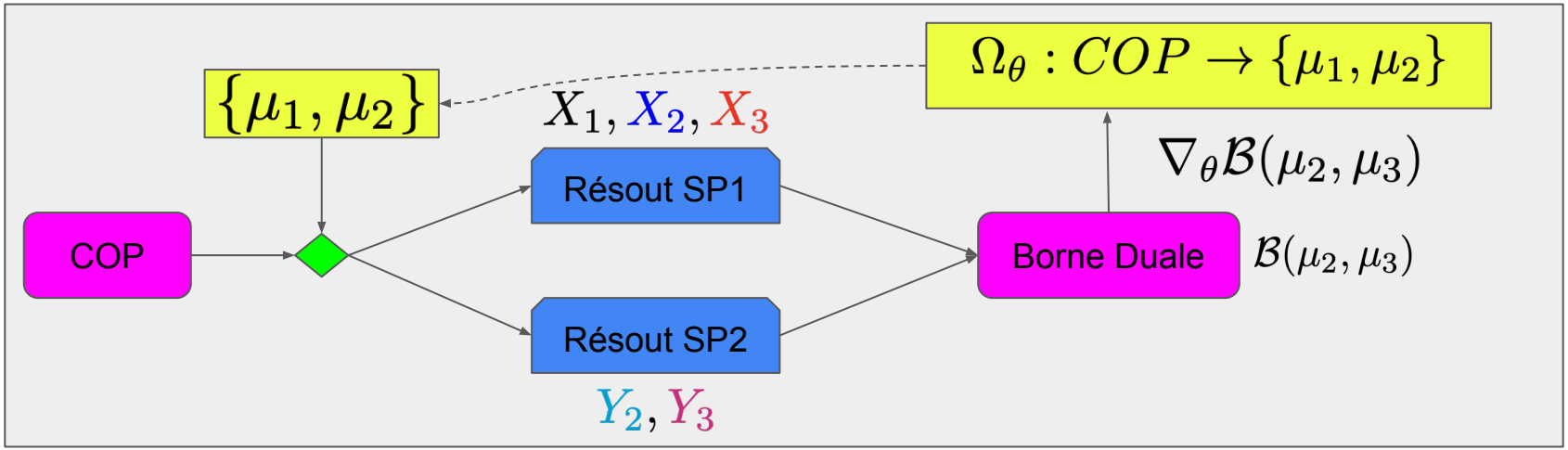
Étape 2 : le modèle est entraîné de bout-en-bout en différenciant la borne



Intuition : le processus d'optimisation est pris en charge durant la phase d'entraînement

Comment calculer le gradient de cette borne

Ω_θ donne $\{\mu_1, \mu_2\}$ mais on veut le mettre à jour selon la qualité de la borne et non $\{\mu_1, \mu_2\}$



$$\nabla_{\Theta}\mathcal{B}(\mu) = \frac{\partial \mathcal{B}(f_{\Theta}(COP)}{\partial \mu} \times \frac{\partial \mu}{\partial \Theta} = \boxed{(X - Y) \times \frac{\partial \mu}{\partial \Theta}}$$

Étape 1 : on utilise la règle de la chaîne pour mettre en évidence les dépendances

Étape 2 : le terme de droite est obtenu par backpropagation du modèle prédictif

Étape 3 : le terme de gauche ré-utilise l'expression du sous-gradient

Entraînement : descente de gradient sur les instances d'entraînement

Résultats

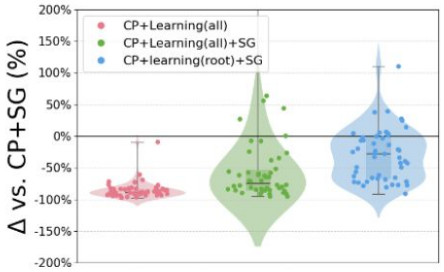
Expériences

Étude de cas : Sac à dos multi-dimensionnel & problème d'ordonnancement des quarts de travail

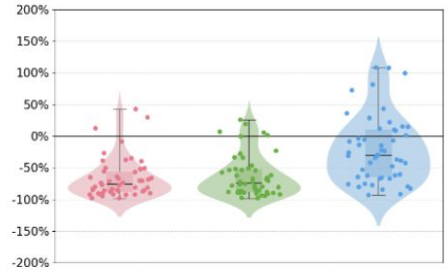
Compétiteurs :

- CP : Approche programmation par contraintes pure
- CP + SG : CP + LD avec multiplicateurs mis à jours itérativement
- CP + Learning(*all*) : CP + LD avec bornes apprises et appliquées à tous les noeuds
- CP + Learning(*all*) + SG : pareil et les bornes obtenues sont améliorées avec SG
- CP + Learning(*root*) + SG : borne apprise appliquée uniquement au noeud racine et utilisée pour initialiser le sous gradient aux autres noeuds

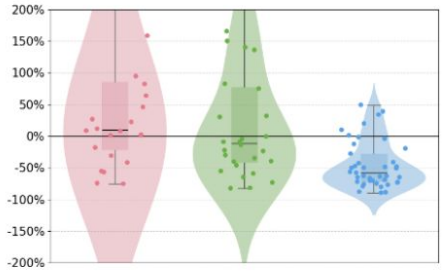
Résultats



(a) MKP avec 30 items



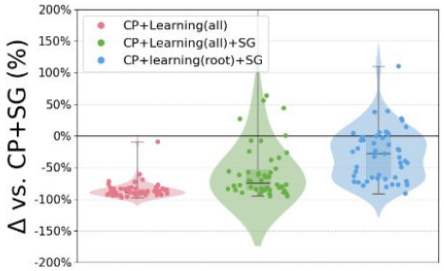
(b) MKP avec 50 items



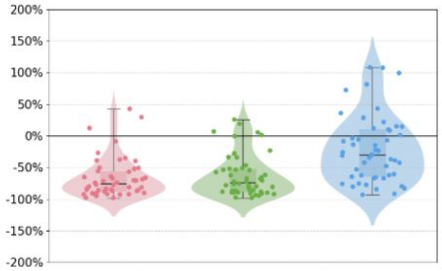
(c) MKP avec 100 items

Chaque point en dessous de 0% indique une réduction du temps avec notre méthode

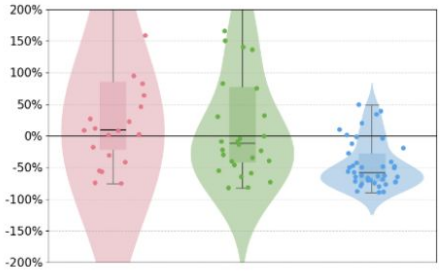
Résultats



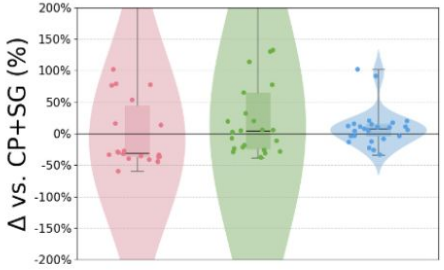
(a) MKP avec 30 items



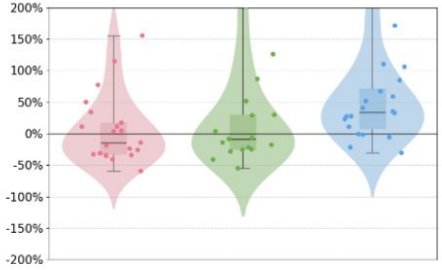
(b) MKP avec 50 items



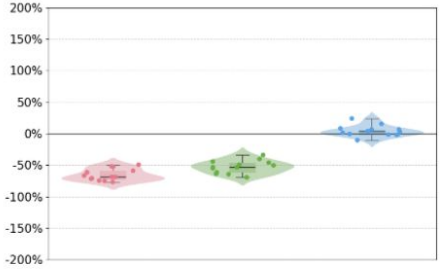
(c) MKP avec 100 items



(d) SSP avec 10 symboles et 20 états



(e) SSP avec 20 symboles et 20 états



(f) SSP avec 10 symboles et 80 états

Chaque point en dessous de 0% indique une réduction du temps avec notre méthode

L'apprentissage a amélioré significativement l'application de la LD à la CP

LD dans CP > Notre Approche > Résultats

Résultats

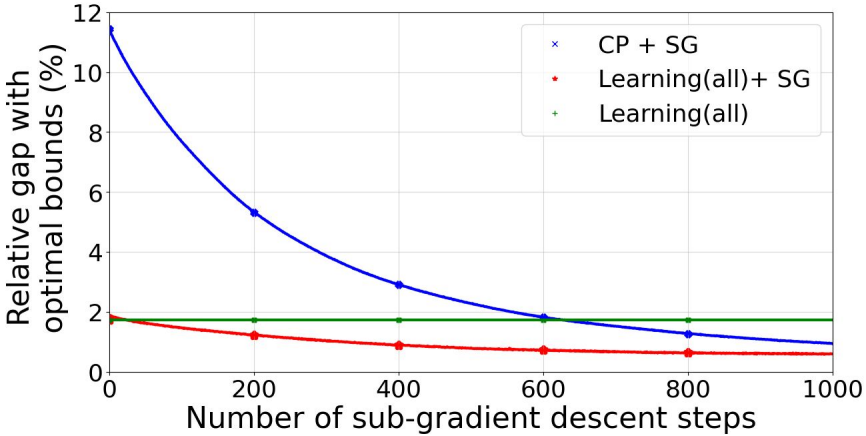
Multi-dimensional Knapsack Problem (MKP)

Approaches	30 items				50 items				100 items			
	No. solved	Time	No. nodes	Gap	No. solved	Time	No. nodes	Gap	No. solved	Time	No. nodes	Gap
CP	50/50	**0.5**	22K	-	30/50	1,100	40M	-	0/50	-	-	-
CP+SG	50/50	19.2	116	2.0%	50/50	158	436	2.0%	41/50	2.4K	4.1K	1.3%
CP+Learning(all)	50/50	2.0	235	6.0%	50/50	**36**	2.6K	3.0%	25/50	2.8K	146K	2.0%
CP+Learning(all)+SG	50/50	6.7	170	2.0%	50/50	40	1.7K	2.0%	29/50	1.6K	33K	1.0%
CP+Learning($root$)+SG	50/50	10.6	**81**	2.0%	50/50	83	**340**	2.0%	**49/50**	**1.1K**	**2.1K**	1.0%

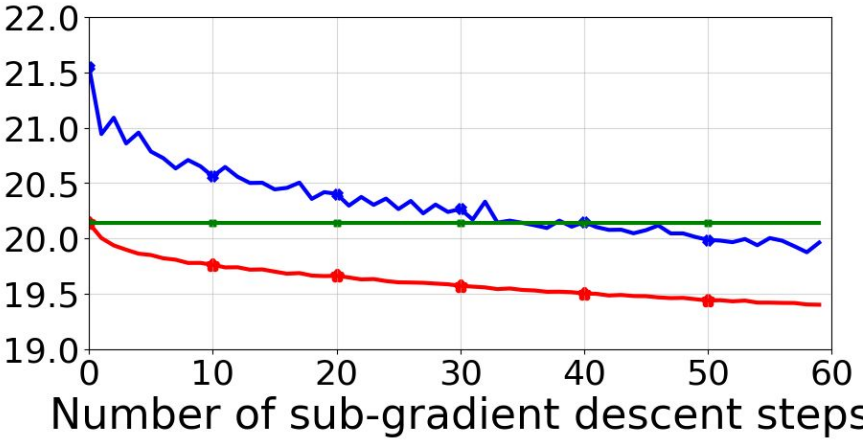
Shift Scheduling Problem (SSP)

Approaches	10 symbols and 20 states				20 symbols and 20 states				10 symbols and 80 states			
	No. solved	Time	No. nodes	Gap	No. solved	Time	No. nodes	Gap	No. solved	Time	No. nodes	Gap
CP+SG	**28/50**	473	2.0K	9.3%	**27/50**	750	**2.9K**	9.8%	13/50	3.3K	**2.5K**	15.8%
CP+Learning(all)	24/50	852	8.8K	10.1%	24/50	**660**	4.7K	10.3%	**20/50**	**1.3K**	3.3K	15.8%
CP+Learning(all)+SG	24/50	880	6.3K	8.2%	22/50	770	4.5K	8.4%	17/50	1.6K	3.1K	15.1%
CP+Learning($root$)+SG	**28/50**	**453**	**1.8K**	8.2%	22/50	1,100	4.5K	8.4%	13/50	3.4K	**2.5K**	15.1%

Résultats



MKP 100 items



SSP 10 vars. 80 states

Évolution de la borne pour les deux configurations les plus compliquées (écart avec la solution optimale)

À retenir

- **Décomposition Lagrangienne** → Bornage automatique (CP)
- **Limitation** → Coût élevé (optimisation sous-gradient)
- **Nouvelle approche** → Apprendre multiplicateur Lagrange → borne duale serrée
- **Apprentissage auto-supervisé** → Modèle GNN → pas de label de borne
- **1ère méthode générique** → Apprentissage de borne duale valide (CP)

papier AAAI 2025

Graph encoding

each node = lagrangian multiplier
MKP : one node per pair (variable, constraint)
n items, d dimension = nxd nodes
edges if share variable or constraint
6 features:

- h_v^1: the index of the related variable.
- h_v^2: the index of the related constraint.
- h_v^3: the profit of the item.
- h_v^4: the weight of the item on the related dimension.
- h_v^5: the ratio of profit to weight.
- h_v^6: the ratio of weight to capacity.

SSP : one node per triplet (variable, constraint, value)
n variables, m constraints, d values = nxmxd nodes
edge if same (variable, value) pair or same constraint (s.t. valid transition)
4 features:

- h_v^1: the index of the related variable.
- h_v^2: the index of the related constraint.
- h_v^3: the index of the related value.
- h_v^4: the profit associated with the triplet.

References

[1] Monique Guignard and Siwhan Kim.
Lagrangean decomposition for integer programming : theory and applications.
RAIRO. Recherche opérationnelle, tome 21, 1987.

[2] 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.

[3] 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.

Guignard et al Lagrangean decomposition for interger programming : theory and applications

Parjadis et al, Learning Lagrangian Multipliers for the Travelling Salesman Problem

Hà et al, General bounding mechanism for constraint programs