Rapport du projet MADMC

Jiahua LI
Zeyu TAO

December 2025

1 Introduction

L'objectif de ce projet est d'implémenter deux approches permettant de générer l'ensemble des points non dominés au sens de Lorenz pour le problème du sac à dos multi-objectifs, puis de comparer et de discuter les performances de ces deux méthodes.

Le problème du sac à dos multi-objectifs peut être formulé comme suit :

$$\begin{aligned} \max \quad & (f_1(x), \cdots, f_n(x)) \\ \text{s.c.} \quad & w^T x \leq W \\ & x_j \in \{0,1\}, \forall j \in \{1, \cdots, m\} \end{aligned} \tag{1}$$

— n objets et m critères.
— $x = (x_1, \cdots, x_m)$, un vecteur d'incidence, $x_j = 1$ si l'objet j est sélectionné, 0 sinon.
— $f(x) = (f_1(x), \cdots, f_n(x))$, une fonction de $\{0,1\}^m$ dans $\mathbb{N}^n$, $f_i(x), \forall i \in \{1, \cdots, n\}$, représente l'évaluation d'une solution x selon le critère i.
— $w = (w_1, \cdots, w_m)$, un vecteur de poids, $w_j, \forall j \in \{1, \cdots, m\}$, représente le poids de l'objet i.
— $W \in \mathbb{N}^*$ représente la capacité maximale du sac à dos.

2 Approche indirecte

Dans cette première partie, nous avons implémenté une première approche, dite *approche indirecte*, pour générer l'ensemble des points non dominés au sens de Lorenz. Elle se divise en deux phases : (i) déterminer tous les points non dominés au sens de Pareto, (ii) à partir des points obtenus dans la première phase, ne garder que ceux qui sont non dominés au sens de Lorenz.

2.1 Programmation dynamique multi-objectifs

Pour déterminer l'ensemble des points non dominés au sens de Pareto, la méthode proposée est la programmation dynamique multi-objectifs, qui dérive directement de la méthode de programmation dynamique classique pour résoudre le problème du sac à dos.

Nous notons $OPT(i,j), \forall i \in \{0,1,\cdots,W\}, \forall j \in \{0,1,\cdots,n\}$ l'ensemble des points Pareto-optimales qui n'utilise que les j premiers objets, dont la somme des poids ne dépasse pas i. La formule de récursion est donnée par :

$$OPT(i,j) = \begin{cases} \vec{0} \in \mathbb{R}^n & \text{si } i = 0 \\ \vec{0} \in \mathbb{R}^n & \text{si } j = 0 \\ OPT(i, j-1) & \text{si } i < w_j \\ M_P\{OPT(i,j-1) \cup \{y + v_j \mid y \in OPT(i - w_j, j-1)\}\} & \text{sinon} \end{cases} \tag{2}$$

où $M_P(X)$ représente l'opérateur pour calculer tous les points Pareto-optimales dans l'ensemble X. L'ensemble des points non dominés au sens de Pareto est donné par : $OPT(W, n)$. L'ensemble des points non dominés au sens de Lorenz est obtenu par : $M_L(OPT(W, n))$, où $M_L(X)$ représente l'opérateur pour calculer tous les points Lorenz-optimales dans l'ensemble X.

Nous constatons que cette première approche ne peut pas être directement utilisée pour calculer l'ensemble des points non dominés au sens de Lorenz, c'est-à-dire en remplaçant simplement l'opérateur $M_P(X)$ par $M_L(X)$ dans l'équation (2). Pour illustrer cela, voici l'exemple suivant à 3 objets $\{o_1, o_2, o_3\}$, 2 critères $\{a_1, a_2\}$ et une capacité maximale de 5 :

	o_1	o_2	o_3
a_1	1	8	9
a_2	9	4	4
w	3	3	2

En résolvant ce problème par la programmation dynamique multi-objectifs en utilisant l'opérateur $M_L(X)$ à la place de $M_P(X)$, nous obtenons $(17, 8)$ comme unique point non dominé au sens de Lorenz, dont l'image dans l'espace de décision est $\{\{o_2, o_3\}\}$. Cependant, il existe un autre point non dominé au sens de Lorenz $(10, 13)$, dont l'image dans l'espace de décision est $\{\{o_1, o_3\}\}$. Mais, comme $(8, 4) \succ_L (1, 9)$, ce dernier ne peut jamais être trouvé par cette procédure, car elle viole le principe d'optimalité de Bellman. Par conséquent, cette procédure **ne peut pas être directement utilisée pour trouver l'ensemble des points non dominés au sens de Lorenz**.

2.2 Réimplémentation en C

Dans un premier temps, nous avons implémenté cette première approche en utilisant Python comme langage de programmation. Nous avons constaté que la performance de cette première version de l'implémentation n'était pas satisfaisante. En particulier, en comparaison avec la seconde approche que nous décrirons par la suite. La différence entre ces deux approches en termes de temps de calcul est très large : plus de 20 minutes pour une instance à 100 objets et 2 critères pour la première approche, contre seulement environ 0.3 secondes pour la seconde approche.

Cette différence s'explique principalement par le choix de l'implémentation : la première approche a été implémentée en Python, tandis que la seconde approche repose sur Gurobi, un solveur MILP principalement écrit en C/C++. Donc afin d'obtenir des résultats plus équitables, nous avons réimplémenté toute la partie de programmation dynamique multi-objectifs en C, en interagissant avec le reste de notre code via un *wrapper* généré par SWIG.

3 Approche directe

Dans cette seconde partie, nous avons implémenté une autre approche, dite *approche directe*. Contrairement à la première approche, elle repose sur une procédure itérative pour générer directement les points non dominés au sens de Lorenz. À chaque itération, elle génère un point non dominé au sens de Lorenz en résolvant un programme linéaire à variables mixtes. Ce nouveau point doit être meilleur de manière stricte pour au moins l'un des composants du vecteur de Lorenz pour chaque point non dominé au sens de Lorenz trouvé préalablement, puis, elle itère jusqu'à ce qu'il ne reste plus aucun point satisfaisant ces conditions.

3.1 Programmation linéaire

La génération de point non dominé au sens de Lorenz est principalement fondée sur la formulation linéaire du modèle OWA en choisissant un vecteur de poids strictement décroissant et positif : $\omega = \omega_1, \cdots, \omega_n$, avec $\omega_1 < \cdots < \omega_n, \quad \omega_i > 0, \forall i \in \{1, \cdots, n\}$.

Nous notons $y^1, \cdots, y^l$ l'ensemble des point non dominés au sens de Lorenz trouvés jusqu'à l'itération l et $L^1, \cdots, L^l$ leurs vecteurs de Lorenz correspondants. Pour trouver la $l+1$-ème point non dominé au sens de Lorenz, s'il existe, nous résolvons le programme linéaire (P) suivant :

$$
\begin{array}{lll}
\max & \displaystyle\sum_{k=1}^{n} \lambda_k (k r_k - \sum_{k=1}^{n} b_i^k) & \\
\text{s.c.} & \displaystyle\sum_{j=1}^{m} w_j x_j \leq W & (1) \\
& f_i = \displaystyle\sum_{j=1}^{m} v_{j,i} x_j \qquad \forall i \in \{1, \cdots, n\} & (2) \\
& r_k - b_i^k \leq f_i \qquad \forall i, k \in \{1, \cdots, n\} & (3) \\
& k r_k - \displaystyle\sum_{i=1}^{n} b_i^k \geq (L_k^s + 1) z_k^s \qquad \forall k \in \{1, \cdots, n\} & (4) \\
& \qquad \forall s \in \{1, \cdots, l\} & \\
& \displaystyle\sum_{k=1}^{n} z_k^s \geq 1 \qquad \forall k \in \{1, \cdots, n\} & (5) \\
& \qquad \forall s \in \{1, \cdots, l\} & \\
& x_j \in \{0,1\} \quad \forall j \in \{1, \cdots, m\} & \\
& f_i \geq 0 \quad \forall i \in \{1, \cdots, n\} & \\
& b_i^k \geq 0 \quad \forall i, k \in \{1, \cdots, n\} & \\
& r_k \in \mathbb{R} \quad \forall k \in \{1, \cdots, n\} & \\
& z_k^s \in \{0,1\} \quad \forall k \in \{1, \cdots, n\}, \forall s \in \{1, \cdots, l\} &
\end{array}
\tag{P}
$$

où $\lambda = (\omega_1 - \omega_2, \cdots, \omega_n - \omega_{n-1}, \omega)$. La contrainte (1) est la contrainte classique du problème du sac à dos. La contrainte (2) affecte l'évaluation de chaque critère de la solution à une variable de décision. La contrainte (3) est la contrainte de linéarisation du modèle OWA. Ces trois premières contraintes garantissent un point non dominé au sens de Lorenz, si le vecteur de poids ω est strictement décroissant et positif. Les contraintes (4) et (5) garantissent que le nouveau point obtenu, s'il existe, doit être meilleur de manière stricte pour au moins l'un des composants du vecteur de Lorenz pour chaque point $y^1, \cdots, y^l$ trouvé.

Nous obtenons les points non dominés au sens de Lorenz en résolvant de manière itérative le programme linéaire (P), et en ajoutant, à chaque itération, les nouvelles contraintes (4) et (5) associées au nouveau point obtenu.

3.2 Amélioration : partie bonus

En effet, cette seconde approche nous permet de déterminer les points non dominés au sens de Lorenz, mais pas tous. Puisque sa résolution repose directement sur le vecteur de Lorenz, notamment via la contrainte (4) pour générer l'ensemble des points, elle ne permet pas de distinguer deux points ayant un vecteur de Lorenz identique. Par exemple les points : $(12, 13, 10)$ et $(12, 10, 13)$.

Pour cela, nous avons adopté l'amélioration suivante, qui repose globalement sur le même principe que (P). Nous notons $Y_s = y^1, \cdots, y^{|Y_s|}$ l'ensemble des points non dominés au sens de Lorenz trouvés jusqu'à l'itération $|Y_s|$, qui partagent le même vecteur de Lorenz L. Pour trouver la $|Y_s|+1$-ème point, nous résolvons le programme linéaire (P') suivant :

$$
\begin{aligned}
\max \quad & \sum_{k=1}^{n} \lambda_k (k r_k - \sum_{k=1}^{n} b_i^k) & & & \\
\text{s.c.} \quad & \sum_{j=1}^{m} w_j x_j \leq W & & & (1) \\
& f_i = \sum_{j=1}^{m} v_{j,i} x_j & & \forall i \in \{1, \cdots, n\} & (2) \\
& r_k - b_i^k \leq f_i & & \forall i, k \in \{1, \cdots, n\} & (3) \\
& k r_k - \sum_{k=1}^{n} b_i^k = L_k & & \forall k \in \{1, \cdots, n\} & (4) \\
& f_i \geq (y_k^s + 1) z_k^s & & \forall k \in \{1, \cdots, n\} & (5) \\
& & & \forall s \in \{1, \cdots, |Y_s|\} & \\
& \sum_{k=1}^{n} z_k^s \geq 1 & & \forall k \in \{1, \cdots, n\} & (6) \\
& & & \forall s \in \{1, \cdots, |Y_s|\} &
\end{aligned}
\qquad (P')
$$

$$
\begin{aligned}
& x_j \in \{0, 1\} \quad \forall j \in \{1, \cdots, m\} \\
& f_i \geq 0 \quad \forall i \in \{1, \cdots, n\} \\
& b_i^k \geq 0 \quad \forall i, k \in \{1, \cdots, n\} \\
& r_k \in \mathbb{R} \quad \forall k \in \{1, \cdots, n\} \\
& z_k^s \in \{0, 1\} \quad \forall k \in \{1, \cdots, n\}, \forall s \in \{1, \cdots, |Y_s|\}
\end{aligned}
$$

Les programmes linéaire (P) et (P') partagent les mêmes contraintes (1), (2) et (3) pour déterminer un point non dominé au sens de Lorenz. La nouvelle contrainte (4) garantit que le nouveau point obtenu, s'il existe, partage le même vecteur de Lorenz L. Les contraintes (5) et (6) garantissent qu'il doit être meilleur de manière stricte pour au moins l'un des critères afin de ne pas être dominé par les autres points trouvés.

4 Expériences

Dans cette dernière partie, nous avons réalisé des comparaisons entre ces deux approches : l'*approche basée sur la programmation dynamique multi-objectifs (indirecte)* et l'*approche basée sur la programmation linéaire (directe)*. Pour ce faire, nous avons utilisé le fichier *"2KP200-TA-0.dat"*, qui contient des données d'une instance du problème du sac à dos multi-objectifs à 200 objets et 6 critères, comme notre source de données principale pour réaliser nos comparaisons.

Pour construire les instances de tests, nous n'avons utilisé que des sous-ensembles de données de cette instance. Par exemple, le TABLE.1 représente les résultats d'une seul exécution sur des instances en prenant que les m premiers objets et les n premiers critères, dont la capacité maximale est égale à la partie entière inférieure de la somme des poids divise par deux : $\lfloor \frac{\sum_{j=1}^{m} w_j}{2} \rfloor$.

Pour la plupart des comparaisons, nous avons utilisé la version non améliorée de l'approche basée sur la programmation linéaire et un vecteur de poids : $\omega = (n, n-1, \cdots, 1)$, qui est défini simplement par une suite arithmétique décroissante. Nous notons que n'importe quelle suite arithmétique décroissante joue un rôle similaire, car ce sont les écarts entre les poids qui influencent vraiment la performance de la résolution, et pas les poids eux-mêmes.

# of objectives (n)			# of items (m)							
			DP-based approach				MILP-based approach			
			20	40	60	80	20	40	60	80
	2	computation time	0.04	0.37	1.62	4.86	0.01	0.05	0.14	0.71
		Pareto size	9	28	43	81	2	4	8	15
		Lorenz size	2	4	8	15	2	4	8	15
	3	computation time	0.08	3.60	59.3	546.	0.02	1.98	1.98	3.18
		Pareto size	17	224	550	1378	2	9	29	31
		Lorenz size	2	9	29	31	2	9	29	31
	4	computation time	0.15	16.2	526.	–	0.03	0.79	1.31	–
		Pareto size	58	597	1827	–	2	12	22	–
		Lorenz size	2	12	22	–	2	12	22	–
	5	computation time	0.55	391.	–	–	0.16	1.07	–	–
		Pareto size	209	3004	–	–	7	11	–	–
		Lorenz size	7	11	–	–	7	11	–	–
	6	computation time	2.19	639	–	–	0.39	9.22	–	–
		Pareto size	472	5332	–	–	9	29	–	–
		Lorenz size	9	29	–	–	9	29	–	–

TABLE 1 – Résultats d'une seule exécution de ces deux approches. Le tableau représente, pour chaque instance à m objets et n critères : le temps de calcul (en secondes), la taille du front de Pareto et le nombre de points non dominé au sens de Lorenz trouvées par chacune des approches.

4.1 Comparaison des performances

Pour comparer des performances entre ces deux approches, nous avons construit, pour chaque paire de configuration m objets et n critères, 10 instances distinctes en sélectionnant aléatoirement m objets (en fixant un seed) et prenant de nouveau les n premiers critères. En total, nous avons exécuté chacune des approches sur 10 seeds distincts pour obtenir des résultats qui soient statistiquement corrects. Les résultats de cette expérience sont illustrés dans la Fig.1.

En effet, nous savons que l'approche basée sur la programmation dynamique multi-objectifs est un algorithme pseudo-polynomial, dont la complexité dépend de deux valeurs numériques : la capacité maximale et la taille du front de Pareto. Nous observons que la Fig.1 nous confirme que le temps de calcul croît fortement avec le nombre d'objets, cette augmentation d'objets impact également la capacité maximale et la taille du front de Pareto. De plus, en comparant les temps de calcul entre les instances à 2 et 3 objectifs, nous observons qu'une augmentation significative en fonction de la taille du front de Pareto (par construction de nos instances de tests, pour toute paire de configuration m objets et n critères, la capacité maximale est identique en fixant m), qui passe de quelques dizaines de secondes pour 2 objectifs à quelques centaines de secondes pour 3 objectifs.

Contrairement à l'approche précédente, l'approche basée sur la programmation linéaire ne dépend pas de la capacité maximale et de la taille du front de Pareto, ou peu, mais elle dépend du nombre de points non dominés au sens de Lorenz. C'est pourquoi nous observons que les temps de calcul de cette approche sont relativement rapides par rapport à autre l'approche, car le nombre de points non dominés au sens de Lorenz est beaucoup plus petit que la taille du front de Pareto. En particulier, nous observons qu'une instance à 70 objets et 3 critères identifiée par un cercle dans la figure en bas à droite, cette instance donne un nombre de points non dominés au sens de Lorenz très important par rapport aux autres instances, et provoque un pic correspondant à la courbe verte gauche.

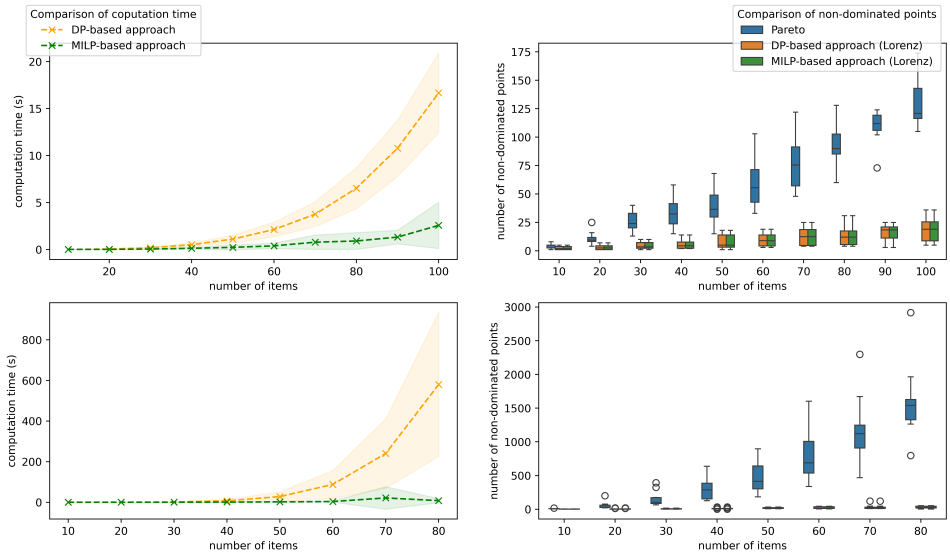

FIGURE 1 – Comparaison entre ces deux approches : *indirecte (DP-based)* et *directe (MILP-based)*. Les figures gauches représentent le temps de calcul moyen (en secondes) pour des instances à 2 et 3 objectifs (ou critères), les zones d'ombre représentent l'écart-type. Les figures droites représentent la distribution des points non dominés trouvés, les boîtes à moustaches représentent les médians et les quartiles.

4.2 Étude du choix du vecteur de poids

Jusqu'ici, nous avons utilisé la suite arithmétique décroissante pour réaliser tous nos tests, mais tous les autres vecteurs de poids strictement décroissants et positifs sont également envisageables. Puisqu'il existe une infinité de vecteur de poids, il est clairement impossible de tous les considérer. Dans cette expérience, nous avons étudié la suite géométrique décroissante : $\omega = (q^{n-1}, \cdots, q^1, q^0)$, avec un ratio $q > 1$ comme une autre alternative, et comparé avec la suite arithmétique.

La construction des instances est identique à celle précédente, à l'exception que nous n'avons utilisé seulement 5 seeds en raison de contraintes de temps. Les résultats de cette expérience sont illustrés dans la Fig.2.

En comparant ces deux types de suites, nous constatons que la suite arithmétique constitue un choix satisfaisant jusqu'à 5 critères et moins de 100 objets, car le temps de calcul en moyen est inférieur à celui de la plupart des suites géométriques. Nous observons que pour des petites instances, les temps de calcul pour les suites géométriques sont assez instables, et lorsque nous augmentons le nombre d'objets ou le nombre de critères, les courbes de temps de calcul convergent vers une forme convexe. Pour des ratios relativement proches de 1, où à l'inverse, pour des valeurs de ratios élevées, les temps de calcul augmentent significativement. Ces deux extrémités correspondent à des vecteurs de poids très équilibrés (écarts très petits entre les poids) et des vecteurs de poids très déséquilibrés (un poids très élevé sur la première composante) respectivement. Ces résultats nous montrent que nous pouvons choisir un ratio q appropriée pour constitue un choix du vecteur de poids satisfaisant, par exemple, des valeurs de $q \in [1.3, 1.5]$.

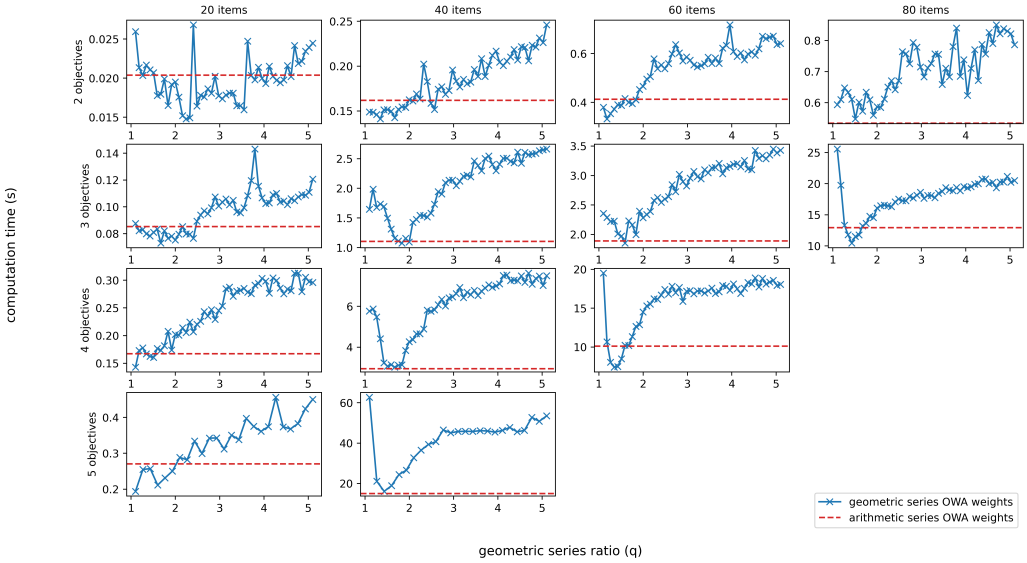

FIGURE 2 – Comparaison entre la suite arithmétique : $\omega = (n, n-1, \cdots, 1)$ et les suites géométriques : $\omega = (q^{n-1}, \cdots, q^1, q^0)$ en variant q. Les figures représentent le temps de calcul moyen (en secondes) pour la suite arithmétique et les suites géométriques.

5 Conclusion

Nous avons implémenté les deux approches : l'approche basée sur la programmation dynamique multi-objectifs (indirecte) et l'approche basée sur la programmation linéaire (directe). Nous avons également implémenté la version améliorée de l'approche basée sur la programmation linéaire pour trouver tous les points non dominés au sens de Lorenz qui partagent le même vecteur de Lorenz. Nous avons réalisé les comparaisons des performances entre ces deux approches, en discutant les critères qui influencent la performance de ces deux approches. Nous avons étudié l'influence du choix du vecteur de poids, en comparant les performances de la suite arithmétique et les suites géométriques.

En résumé, nous concluons que l'approche basée sur la programmation linéaire est généralement plus performante que l'approche basée sur la programmation dynamique multi-objectifs en termes de temps de calcul et de consommation de mémoire, mais dans des situations, où la taille du front de Pareto est raisonnable, ou lorsque le nombre de points non dominés au sens de Lorenz est relativement grand par rapport à la taille du front de Pareto, il peut être envisageable d'utiliser cette approche à la place de l'autre.