Projet MADMC

Génération de l'ensemble des points non dominés au sens de Lorenz pour le problème du sac à dos multi-objectifs

Guillaume Lebreton - 21109566

2025 - 2026

1 Introduction

Dans ce projet, on s'intéresse à la génération de l'ensemble des solutions non dominées au sens de Lorenz pour le problème du sac à dos multi-objectifs.

Deux méthodes de résolution sont étudiées. La première est une méthode indirecte, qui consiste à générer l'ensemble des solutions Pareto non dominées par programmation dynamique, puis à appliquer un filtrage au sens de Lorenz. La seconde est une méthode directe, basée sur la résolution itérative d'un problème d'optimisation intégrant une fonction d'agrégation de type OWA, permettant d'énumérer directement les solutions Lorenz non dominées.

1.1 Strucure

L'ensemble du projet à été implementé en python. Le code a été structuré en différent fichier :

- **instance.py** : Défini une classe *Instance* qui représente une instance du problème (n, p, poids, valeurs, capacité) et une méthode pour créer une instance à partir d'un fichier dat ou aléatoirement.
- **indirecte.py** : Implémente la méthode indirecte, $methode\_indirecte(instance)$ renvoie la liste des points Pareto et Lorenz non dominés.
- **direct.py** : Implémente la méthode direct, $enumerate\_lorenz(instance, omega)$ renvoie la liste des points Lorenz non dominés et les vecteurs de Lorenz.
- **test.py** : Exécute et compare les 2 méthodes sur une instance donnée.
- **omega_test.py** : Permet de tester l'influence du jeu de poids ω dans la méthode directe.
- **comparaion.py** : Permet de comparer les 2 méthodes sur plusieurs instance

L'intégralité du code est sur github

2 Méthode indirecte (indirecte.py)

La méthode indirecte est implémentée sous la forme d'une procédure en deux étapes.

Dans un premier temps, la fonction `pareto_dp(instance)` met en œuvre une programmation dynamique afin de générer l'ensemble des vecteurs objectifs Pareto non dominés. Pour chaque capacité, un front de Pareto est maintenu et mis à jour de manière incrémentale lors de l'ajout de chaque objet. Une fois tous les objets traités, les solutions obtenues pour les différentes capacités sont filtrées afin de construire le front Pareto global.

Dans un second temps, la fonction `lorenz_filter(points)` applique un filtrage au sens de Lorenz sur l'ensemble des vecteurs objectifs Pareto non dominés. Les vecteurs de Lorenz associés aux solutions sont calculés à l'aide de la fonction `lorenz_vector`, et seuls les points correspondant à des vecteurs de Lorenz non dominés sont conservés.

La procédure `methode_indirecte(instance)` combine les 2 fonctions et retourne à la fois l'ensemble des vecteurs objectifs Pareto non dominés et l'ensemble des vecteurs objectifs Lorenz non dominés correspondants à partir d'une instance du problème.

3 Méthode directe (direct.py)

La méthode directe à été implémenter sous la forme d'une fonction chargée d'énumérer les solutions non dominées d'une instance du problème. La fonction **enumerate_lorenz(instance, omega)**, commence par calculer les λ pour l'OA à partir des poids ω donnés et initialise 2 listes vides pour stocker les vecteurs de Lorenz solutions et leurs vecteurs objectifs associés.

À chaque itération, la fonction construit et résout un modèle OWA à l'aide du solveur **Gurobi**. La solution obtenue permet de calculer un vecteur objectif ainsi que le vecteur de Lorenz associé, qui sont ajoutés à l'ensemble des solutions trouvées. Après chaque résolution, des contraintes d'amélioration sont ajoutées au modèle afin d'empêcher la réapparition des solutions déjà obtenues. Ces contraintes garantissent que la nouvelle solution améliore strictement au moins une composante par rapport aux solutions précédentes. Le processus est répété jusqu'à ce qu'aucune solution admissible supplémentaire ne puisse être trouvée.

La fonction retourne l'ensemble des solutions Lorenz non dominées, sous la forme de leurs vecteurs objectifs et des vecteurs de Lorenz correspondants.

Cette méthode garantit l'obtention de tous les vecteurs de Lorenz non dominés mais pas nécessairement tous les points Lorenz non dominés car 2 points peuvent avoir le même vecteur de Lorenz. Pour cela il faudrait exclure les solutions précédentes au niveau des vecteurs objectifs, et non uniquement au niveau des vecteurs de Lorenz. Dans le fichier **direct.py** on a gardé la méthode naïve et ne récupérons donc pas forcément tous les points Lorenz non dominés.

4 Résultats et comparaison

4.1 Analyse de l'influence de ω

La méthode directe repose sur l'optimisation répétée d'une fonction d'agrégation OWA, dont les poids influencent fortement la difficulté des programmes linéaires résolus. Avant de comparer les méthodes directe et indirecte, on va donc étudier l'influence du choix du jeu de poids ω sur les temps de calcul.

4.1.1 Génération des jeux de poids

Afin de générer des jeux de poids de répartition différente on va utiliser une méthode exponentielle. Pour un nombre d'objectifs p et un paramètre $\lambda > 0$, les poids sont définis par:

$$\omega_k(\lambda) = \frac{e^{-\lambda k}}{\sum_{i=1}^{p} e^{-\lambda i}}, \quad k = 1, \ldots, p.$$

Des valeurs faibles de λ conduisent à des poids presque uniformes, tandis que des valeurs élevées privilégient fortement les premières composantes.

4.1.2 Protocole expérimental

Les expériences ont été réalisées pour différents nombres d'objectifs ($p = 2, 4, 6$) et différentes tailles d'instances. Afin de maintenir des temps de calcul raisonnables, des tailles plus réduites ont été considérées pour les valeurs élevées de p. Les valeurs du paramètre λ testées sont :

$$\lambda \in \{0.01,\ 0.1,\ 0.25,\ 0.4,\ 0.5,\ 0.6,\ 0.75,\ 1\}.$$

Les jeux de poids OWA complets générés pour les différentes valeurs de λ sont les suivants:

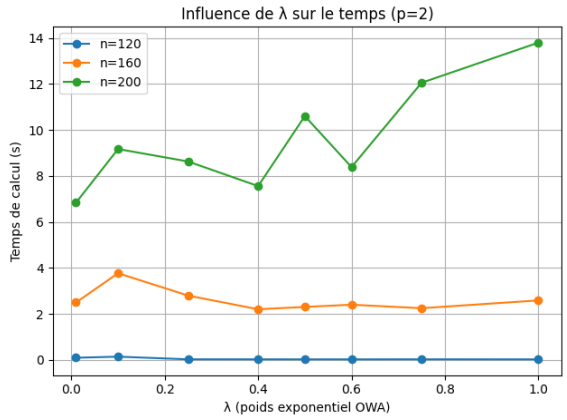
λ	$p = 2$	$p = 4$	$p = 6$
0.01	[0.50, 0.50]	[0.25, 0.25, 0.25, 0.25]	[0.17, 0.17, 0.17, 0.17, 0.16, 0.16]
0.10	[0.53, 0.47]	[0.29, 0.26, 0.24, 0.21]	[0.21, 0.19, 0.17, 0.16, 0.14, 0.13]
0.25	[0.56, 0.44]	[0.35, 0.27, 0.21, 0.17]	[0.29, 0.22, 0.17, 0.13, 0.11, 0.08]
0.40	[0.60, 0.40]	[0.41, 0.28, 0.19, 0.12]	[0.36, 0.24, 0.16, 0.11, 0.07, 0.05]
0.50	[0.62, 0.38]	[0.46, 0.28, 0.17, 0.10]	[0.41, 0.25, 0.15, 0.09, 0.06, 0.03]
0.60	[0.65, 0.35]	[0.50, 0.27, 0.15, 0.08]	[0.46, 0.26, 0.14, 0.08, 0.04, 0.02]
0.75	[0.68, 0.32]	[0.56, 0.26, 0.12, 0.06]	[0.53, 0.25, 0.12, 0.06, 0.03, 0.01]
1.00	[0.73, 0.27]	[0.64, 0.24, 0.09, 0.03]	[0.63, 0.23, 0.09, 0.03, 0.01, 0.00]

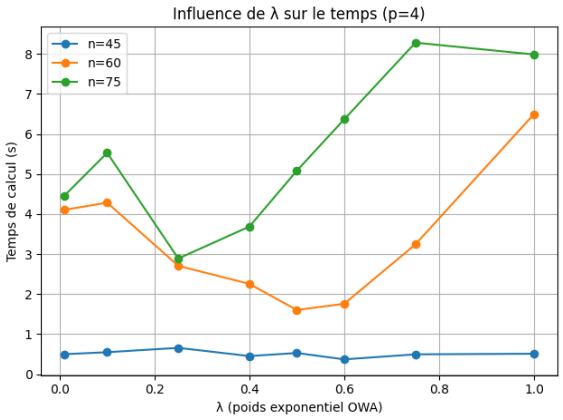
Table 1: Vecteurs de poids ω (arrondis au centième) en fonction de λ et de la dimension p

Pour chaque couple (p, n), le temps total nécessaire à l'énumération des vecteurs de Lorenz non dominés par la méthode directe est mesuré sur une moyenne de 3 runs. Les résultats sont affichés dans le terminal mais également sauvegardés dans un CSV et tracés à l'aide de matplotlib.

4.1.3 Résultats expérimentaux

Les figures suivantes présentent l'évolution du temps de calcul en fonction du paramètre λ pour les différentes tailles d'instances et de nombres d'objectifs. Le tableau récapitule pour chaque couple (p,n) le meilleur λ aisni que le ω associé.





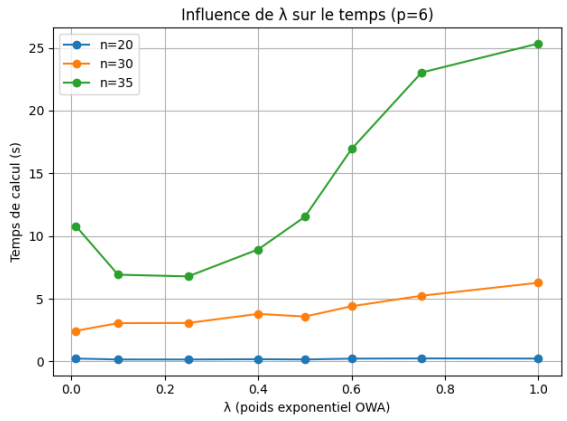


Tableau des λ optimaux et ω associés

p	n	λ^*	ω^*
2	120	0.60	[0.646, 0.354]
	160	0.40	[0.599, 0.401]
	200	0.01	[0.502, 0.498]
4	45	0.60	[0.496, 0.272, 0.149, 0.082]
	60	0.50	[0.455, 0.276, 0.167, 0.102]
	75	0.25	[0.350, 0.273, 0.212, 0.165]
6	20	0.25	[0.285, 0.222, 0.173, 0.134, 0.105, 0.082]
	30	0.01	[0.171, 0.169, 0.167, 0.166, 0.164, 0.163]
	35	0.25	[0.285, 0.222, 0.173, 0.134, 0.105, 0.082]

Figure 1: Influence du paramètre λ sur le temps de calcul

4.1.4 Analyse et interprétation

Les résultats montrent qu'il n'existe pas de valeur optimale universelle du paramètre λ. La valeur minimisant le temps de calcul dépend à la fois du nombre d'objectifs et de la taille de l'instance. Néanmoins, les valeurs intermédiaires de λ, typiquement comprises entre 0.25 et 0.5, conduisent systématiquement à

des bons temps de calcul.

Les valeurs très faibles de λ, tendent à générer un nombre élevé d'itérations, tandis que des valeurs élevées de λ rendent les programmes linéaires plus difficiles à résoudre. Ces observations mettent en évidence l'existence d'un compromis entre sélectivité de l'agrégation et difficulté de résolution.

4.1.5 Choix du jeu de poids retenu

Au vu de ces résultats, la valeur $\lambda = 0.5$ est retenue pour la suite de l'étude. Bien qu'elle ne soit pas toujours strictement optimale, cette valeur offre un compromis robuste, conduisant à des performances proches du minimum observé sur l'ensemble des configurations testées.

4.2 Comparaison des 2 méthodes

4.2.1 Protocole expérimental

Afin de comparer les 2 méthodes on effectue un benchmark des temps d'exécution, nombre de points Pareto non dominés (pour la méthode indirecte) et nombre de points Lorenz non dominés retournés par les 2 méthodes sur des instances de p et n différents. Le fichier **comparaison.py** permet d'effectuer ce benchmark et de stocker les résultats dans un fichier csv.

Pour analyser les données du csv le fichier **analyse_results.py** propose différentes fonctions d'affichage des résultats :

- *plot_temps_exec* : trace les courbes d'éxécution des 2 méthodes en fonction de n pour chaque nombre d'objectifs p
- *plot_points_count* : trace le nombre de points Pareto non dominés et Lorenz non dominés obtenus avec les 2 méthodes en fonction de n pour chaque p

Ici on a effectué le benchmark suivant:

p	n
2	[50, 75, 100, 125, 150]
3	[30, 40, 50, 60, 65]
4	[20, 30, 40, 45, 50]
5	[15, 20, 30, 35, 40]
6	[10, 15, 20, 25, 30]

4.2.2 Résultats

Voici les graphiques obtenus pour les valeurs de p = 2,4,6
Les résultats pour les autres valeurs de p confirment les mêmes tendances et ne sont pas présentés ici pour des raisons de lisibilité. On peut retouver tous les graphiques de comparaison des 2 méthodes dans le dossier *Resultats/plots/*.

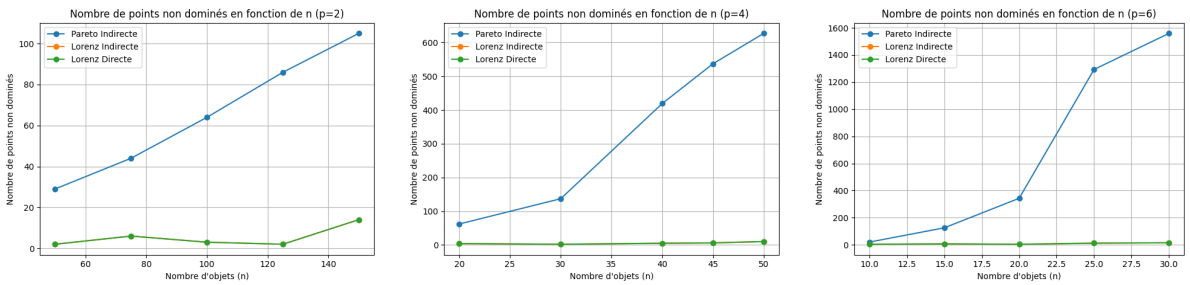


Figure 2: Nombres de points Pareto/Lorenz non dominés en fonction de n

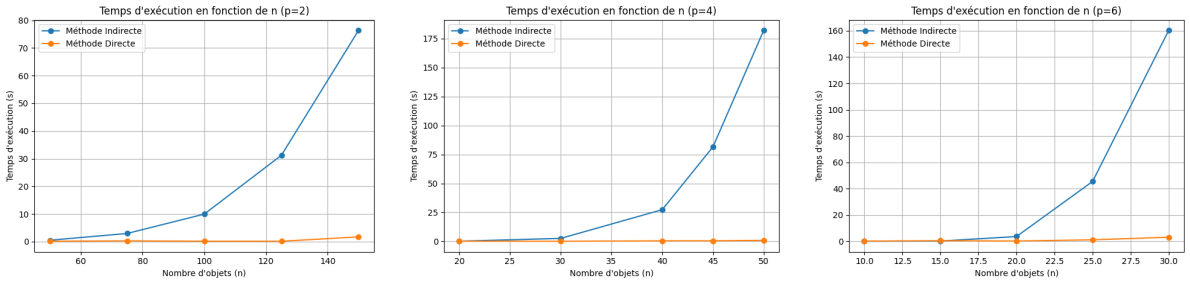


Figure 3: Temps d'exécutions des 2 méthodes en fonction de n

4.2.3 Analyse et interprétation

Dans un premier temps, on observe que **le nombre de solutions** Pareto non dominées croît très rapidement avec le nombre d'objets n, phénomène qui s'amplifie lorsque le nombre d'objectifs p augmente. Cette explosion du front de Pareto est une propriété classique des problèmes multi-objectifs et explique la difficulté de la méthode indirecte à gérer de grandes instances.

À l'inverse, le nombre de solutions non dominées au sens de Lorenz reste relativement limité, même pour des instances de taille plus importante. Le critère de Lorenz apparaît ainsi comme nettement plus sélectif que la dominance de Pareto, en éliminant un grand nombre de solutions déséquilibrées.

Les résultats obtenus par la méthode indirecte et par la méthode directe sont cohérents, les deux approches conduisant souvent aux mêmes ensembles de solutions Lorenz non dominées. Même si dans certains cas, plusieurs solutions distinctes peuvent partager le même vecteur de Lorenz, alors ne retourner qu'une solution représentative. Toutefois, ce phénomène devient rare lorsque la taille des instances augmente. On obtient quasiment toujours le même nombre de solutions Lorenz optimales avec les deux méthodes.

L'analyse des temps de calcul montre que la méthode indirecte devient rapidement coûteuse lorsque la taille des instances augmente. Cette augmentation est directement liée à la croissance du nombre de solutions Pareto intermédiaires manipulées lors de la programmation dynamique, ce qui entraîne un surcoût important lors des opérations de filtrage et de mise à jour des fronts.

À l'inverse, la méthode directe présente des temps de calcul beaucoup plus stables et reste performante sur des instances plus grandes. Cette méthode ne nécessite pas l'énumération complète du front de Pareto et exploite efficacement la résolution de problèmes d'optimisation, ce qui explique sa meilleure capacité à traiter des instances de grande dimension.

Ces résultats montrent que, bien que la méthode indirecte soit conceptuellement simple et permette d'obtenir l'ensemble des solutions Pareto non dominées, son coût computationnel limite ses performances. La méthode directe apparaît ainsi comme une approche plus adaptée pour la génération des solutions Lorenz non dominées lorsque le nombre d'objectifs ou la taille des instances augmente.

En pratique, la méthode indirecte est pertinente pour des instances de petite taille, tandis que la méthode directe s'avère plus robuste et plus efficace pour des problèmes multi-objectifs de plus grande dimension.

5 Conclusion

Dans ce projet, deux approches pour la génération des solutions Lorenz non dominées ont été implémentées et comparées. Les résultats expérimentaux montrent que la méthode indirecte, bien qu'efficace sur de petites instances, souffre d'une explosion combinatoire du front de Pareto. À l'inverse, la méthode directe offre de meilleures performances et une meilleure scalabilité, ce qui en fait une approche plus adaptée aux problèmes multi-objectifs de grande dimension.