FACULTÉ DES SCIENCES
UNIVERSITÉ D'ANGERS

LERIA

Rapport de stage : Implémentation C++ générique et comparaison expérimentale d'algorithmes de sélection de points dans un front de Pareto

MASTER 1 Informatique

Étudiant :
Habib KHTEIRA

Encadrant :
Nicolas DUPIN

June 19, 2025

Contents

Remerciements

Je tiens à exprimer ma profonde gratitude à mon encadrant, Monsieur Nicolas DUPIN, pour son soutien et ses conseils précieux tout au long de mon stage.

1 Introduction

Dans le cadre de mes études de Master en informatique, j'ai eu l'opportunité de réaliser un stage de recherche de deux mois au sein du laboratoire LERIA (Laboratoire d'Études et de Recherche en Informatique d'Angers), sous la direction de Nicolas Dupin, enseignant-chercheur à l'Université d'Angers. Ce stage s'est inscrit dans le domaine de l'**optimisation combinatoire multiobjectif**, un champ qui traite de problèmes où plusieurs critères, souvent contradictoires, doivent être optimisés simultanément. Ce type de problème est très courant dans la vie réelle ; par exemple, le problème du plus court chemin peut être formulé en prenant en compte plusieurs objectifs, comme le coût (consommation de carburant, péages, etc.) et la distance (kilomètres parcourus) [8].

Dans ce contexte, les solutions optimales ne se réduisent pas à un unique optimum, mais forment un **ensemble de compromis** appelés **solutions non dominées**. Un point est dit dominé s'il existe une autre solution meilleure sur tous les objectifs, et strictement meilleure sur au moins l'un d'eux. L'ensemble des solutions non dominées constitue le **front de Pareto**, qui représente les meilleures alternatives selon les objectifs considérés.

Cependant, ces fronts peuvent contenir un grand nombre de points (parfois plusieurs centaines ou milliers), rendant leur exploitation directe difficile. Il devient alors nécessaire de sélectionner un **sous-ensemble représentatif** de ces solutions, qui préserve à la fois leur diversité et leur répartition dans l'espace objectif. Plusieurs travaux ont ainsi proposé des métriques pour évaluer la qualité de ces sous-ensembles, notamment dans une perspective de représentation fidèle du front initial [10].

Le sujet de mon stage portait précisément sur cette problématique : concevoir et développer une bibliothèque C++ modulaire pour l'implémentation générique d'algorithmes de **sélection de points** dans un front de Pareto et de realiser des **analyse expérimentale et comparative**. Plus particulièrement, mon travail s'est concentré sur diverses variantes du problème de **p-dispersion**, dont certaines peuvent être résolues exactement à l'aide de techniques de programmation dynamique.

2 Définition des problèmes de sélection

La sélection d'un sous-ensemble représentatif de points à partir d'un front de Pareto est un problème crucial en optimisation multiobjectif. L'objectif est de réduire la complexité d'un grand ensemble de solutions tout en conservant leur diversité et leur représentativité. Deux grandes familles d'approches existent pour aborder ce problème : les méthodes basées sur la p-dispersion, qui visent à maximiser la dispersion géométrique des points choisis, et celles basées sur la mesure d'hypervolume, qui évaluent la "qualité" d'un ensemble de points en fonction de l'espace qu'ils dominent.

2.1 Table de notations

d	Dimension de l'espace objectif ($d \geq 2$)
n	Nombre total de points du front
p	Taille du sous-ensemble à sélectionner ($p \ll n$)
$E = \{x_i\}_{i=1}^n$	Ensemble de solutions non dominées
$x_i \in \mathbb{R}^d$	Un point du front de Pareto
$\alpha > 0$	Exposant utilisé pour pondérer les distances
$d(x, y)$	Distance métrique définie sur $\mathbb{R}^d$
$(x_{\text{ref}}, y_{\text{ref}})$	Point de référence pour le calcul d'hypervolume (en 2D)

Table 1: Notations utilisées dans ce rapport

2.2 Problèmes de sélection : formulation générale

Soit $E = \{x_1, \ldots, x_n\} \subset \mathbb{R}^d$ un ensemble de solutions non dominées dans un espace objectif de dimension $d \geq 2$, où tous les objectifs sont à minimiser.

Relation de dominance dans $\mathbb{R}^d$: Pour deux points $x, y \in \mathbb{R}^d$, on dit que x domine y (noté $x \prec y$) si et seulement si :

$$x \prec y \iff (\forall i \in \{1, \ldots, d\},\ x_i \leq y_i) \wedge (\exists j \in \{1, \ldots, d\},\ x_j < y_j)$$

2.2.1 p-dispersion généralisée

Différentes variantes ont été proposées[7], reposant sur une distance $d(x_i, x_j)$ et une puissance $\alpha > 0$:

- **Max-Min (PMm)** : maximiser la distance minimale entre les paires

$$\max_{\{x_1,\ldots,x_p\} \subset E} \min_{i<j} d(x_i, x_j)$$

- **Max-Sum (PMS)** : maximiser la somme des distances entre toutes les paires

$$\max_{\{x_1,\ldots,x_p\} \subset E} \sum_{i<j} d(x_i, x_j)^\alpha$$

- **Max-Sum-Min (PMSm)** : pour chaque point, prendre la plus petite distance à un autre point, puis sommer

$$\max_{\{x_1,\dots,x_p\}\subset E} \sum_{i=1}^{p} \min_{j\neq i} d(x_i, x_j)^\alpha$$

- **Max-Min-Sum (PMmS)** : minimiser, parmi les points, la somme de distances aux autres, puis maximiser

$$\max_{\{x_1,\dots,x_p\}\subset E} \min_{i=1,\dots,p} \sum_{j\neq i} d(x_i, x_j)^\alpha$$

Ces variantes sont en général **NP-difficiles**, mais des solutions exactes ou approchées peuvent être obtenues en 2D grâce à la structure du front.

2.2.2 Hypervolume (cas général)

Une alternative aux critères purement géométriques de dispersion consiste à maximiser l'**hypervolume** dominé par un sous-ensemble de p points sélectionnés sur un front de Pareto. Cette approche repose sur une idée intuitive : un ensemble de points est jugé représentatif s'il domine un volume important de l'espace objectif.

L'hypervolume mesure donc le volume de la région dominée par les points sélectionnés, relativement à un point de référence supposé dominé par tous les points du front. Cette mesure est cohérente avec la dominance de Pareto et permet d'intégrer des préférences.

2.3 Structure des fronts 2D, distances métriques et implications spécifiques

Dans ce travail, on suppose que le **front de Pareto bidimensionnel** (PF) est donné explicitement sous la forme d'un ensemble de n points $E = \{x_i\}_{i=1}^n \subset \mathbb{R}^2$, chacun représentant une solution non dominée dans un espace à deux objectifs minimisés[2].

Relations de dominance et tri du front : Pour tout couple de points $y = (y_1, y_2)$, $z = (z_1, z_2) \in \mathbb{R}^2$, on définit :

$$\begin{aligned} y \prec z &\iff y_1 < z_1 \text{ et } y_2 > z_2 && \text{(dominance stricte)} \\ y \preceq z &\iff y \prec z \text{ ou } y = z && \text{(dominance large)} \\ y \; I \; z &\iff y \prec z \text{ ou } z \prec y && \text{(incomparabilité stricte)} \end{aligned}$$

Lemme 1: Soit $E = \{x_i\}_{i=1}^n \subset \mathbb{R}^2$ un front de Pareto bidimensionnel. Il est possible de réordonner les points de E en $\mathcal{O}(n \log n)$ de façon à garantir :

$$\forall i_1 < i_2, \quad x_{i_1} \prec x_{i_2} \quad \text{et} \quad \forall i_1 \leq i_2, \quad x_{i_1} \preceq x_{i_2}$$

Cela signifie que le front peut être vu comme une **séquence strictement ordonnée** de gauche à droite (selon x) et de haut en bas (selon y), les deux objectifs étant minimisés. Ce comportement est illustré à la figure 2, où les relations I et $\prec$ sont représentées graphiquement. Le lemme est démontré formellement dans [**?**]

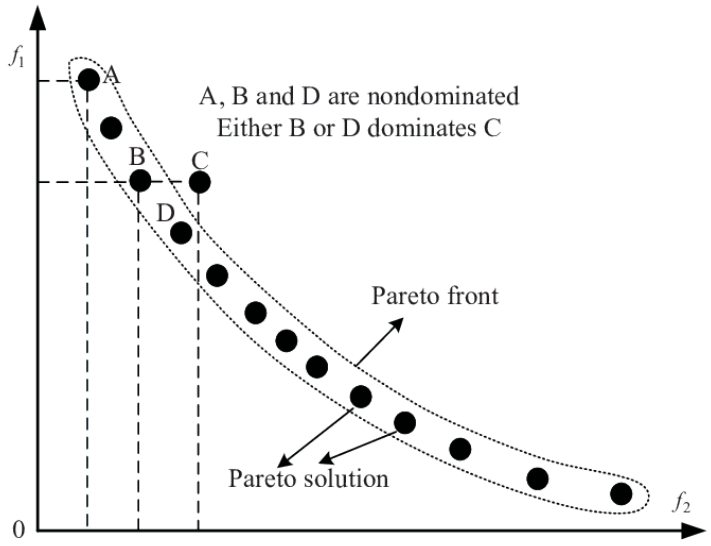


Figure 1: Exemple illustratif d'un front de Pareto pour un problème biobjectif [3]

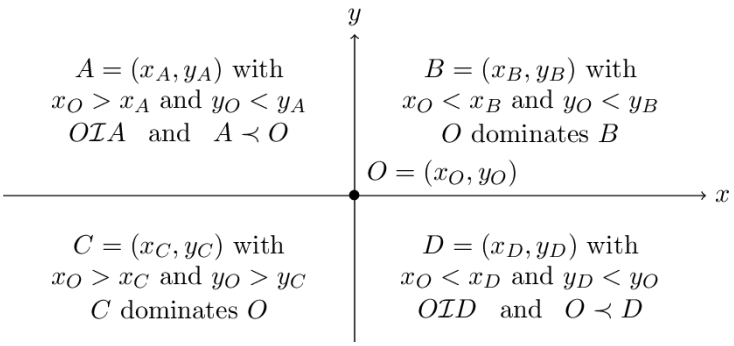


Figure 2: Illustration des relations I et $\prec$, extraite de [2].

Distances métriques utilisées : Les mesures de dispersion reposent sur des distances définies sur $\mathbb{R}^2$. Deux distances sont fréquemment utilisées :

- **Distance de Minkowski d'ordre** $r \geq 1$:

$$d_r(x, y) = (|x_1 - y_1|^r + |x_2 - y_2|^r)^{\frac{1}{r}}$$

- **Distance de Chebyshev (ou** L_∞**)** :

$$d_\infty(x, y) = \max(|x_1 - y_1|, |x_2 - y_2|)$$

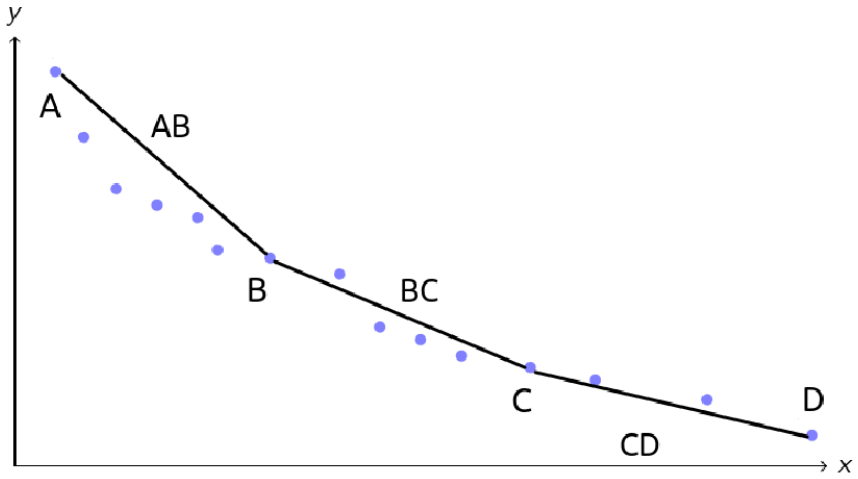
Une puissance $\alpha > 0$ peut être appliquée à ces distances afin de moduler l'importance des écarts.

2.3.1 Variante dédiée : Max-Sum-Neighbor (PMSN)

La structure ordonnée des fronts 2D permet l'introduction d'une variante spécifique :

$$\max_{1 \leq i_1 < \cdots < i_p \leq n} \sum_{j=1}^{p-1} d(x_{i_j}, x_{i_{j+1}})^\alpha$$

Cette variante, appelée **Max-Sum-Neighbor (PMSN)**, maximise les distances entre points successifs sur un front trié, garantissant une bonne répartition le long du front.



Variant		Dispersion of A,B,C,D
Mm	:	$\min(AB^\alpha,BC^\alpha,CD^\alpha) = \min(AB,BC,CD)^\alpha$
MS	:	$AB^\alpha+AC^\alpha+AD^\alpha+BC^\alpha+BD^\alpha+CD^\alpha$
MSN	:	$AB^\alpha+BC^\alpha+CD^\alpha$
MSm	:	$AB^\alpha+\min(AB^\alpha,BC^\alpha)+\min(BC^\alpha,CD^\alpha)+CD^\alpha$
MmS	:	$\min(AB^\alpha+AC^\alpha+AD^\alpha,\ AB^\alpha+BC^\alpha+BD^\alpha,$ $AC^\alpha+BC^\alpha+CD^\alpha,\ AD^\alpha+BD^\alpha+CD^\alpha)$

Figure 3: Exemple illustratif des différentes variantes de dispersion[2]

2.3.2 Hypervolume dans le cas bidimensionnel

Dans le cas biobjectif ($d = 2$), l'hypervolume peut être calculé efficacement grâce à la structure ordonnée du front. Soit $S = \{x_{i_1}, \ldots, x_{i_p}\}$ un sous-ensemble de points triés selon la première composante (croissante en x_1), et soit $(x_{\text{ref}}, y_{\text{ref}})$ un point de référence dominé par tous les points de S. L'hypervolume est alors donné par :

$$HV(S) = \sum_{j=1}^{p-1} (y_{\text{ref}} - y_{i_j}) \cdot (x_{i_{j+1}} - x_{i_j})$$

Cette formule permet un calcul exact en temps linéaire après tri, et exploite la structure stricte du front dans $\mathbb{R}^2$, où les points peuvent être ordonnés de manière croissante en x_1 et décroissante en x_2.

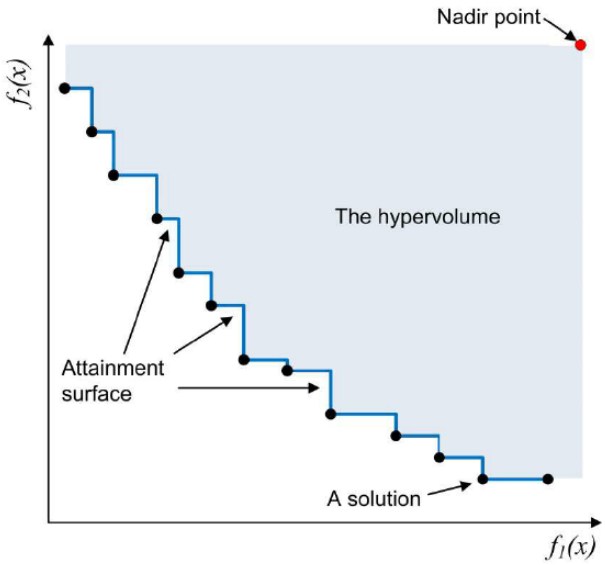


Figure 4: Exemple illustratif d'un hypervolume [6]

3 État de l'art

3.1 Algorithmes pour la sélection de sous-ensembles sur front de Pareto

Les problèmes de sélection de sous-ensembles basés sur les critères de *p-dispersion* (PD) et de *Hypervolume Subset Selection* (HSS) peuvent être abordés par des méthodes exactes et heuristiques.

Une méthode de base pour résoudre ces problèmes consiste à tester toutes les combinaisons possibles de points, ce qui conduit à un algorithme bruteforce de complexité en $O(n^p)$. Cette approche devient rapidement inapplicable pour des valeurs élevées de p ou pour des variantes plus complexes des problèmes.

Pour pallier ces limitations, plusieurs travaux ont introduit des algorithmes exacts ou approchés à complexité polynomiale, notamment pour certaines variantes de la p-dispersion et pour la sélection basée sur l'hypervolume pondéré. Ces méthodes exploitent des propriétés spécifiques des problèmes pour réduire drastiquement l'espace de recherche.

Dans la suite, nous nous intéresserons particulièrement aux algorithmes polynomiaux adaptés aux variantes du problème de p-dispersion [2] et à la sélection par hypervolume pondéré en deux dimensions[5].

3.1.1 Algorithme dynamique pour le problème p-dispersion-MSN

Le problème p-dispersion-MSN (*Max-Sum-Neighbor*) vise à sélectionner un sous-ensemble de p points dans un front de Pareto 2D tel que la somme des distances entre voisins consécutifs dans l'ensemble sélectionné soit maximisée.

Une approche efficace basée sur la programmation dynamique a été proposée dans [2]. Cet algorithme repose sur une matrice C de dimension $(p-1) \times n$, où chaque entrée $C_{k,i}$ représente le meilleur coût obtenu en sélectionnant k points parmi les i premiers, en imposant que le k-ième point sélectionné soit x_i. L'algorithme procède par récurrence

croissante sur k, avec des transitions effectuées en maximisant sur les indices $j \in [k-1, i-1]$.

Sa complexité en temps est $\mathcal{O}(pn^2)$, et en espace $\mathcal{O}(pn)$. Il permet également de reconstruire efficacement l'ensemble optimal des indices sélectionnés.

Algorithm 1 p-dispersion-MSN dans un front de Pareto 2D avec $p \geq 3$

Require: n points d'un front de Pareto 2D, $E = \{x_i\}_{i \in [\![1,n]\!]}$; un entier $p \in [\![3, n]\!]$
1: Réindexer E selon l'ordre du Lemme 1
2: Initialiser la matrice C avec $C_{k,i} := 0$ pour tout $i \in [\![1, n]\!]$, $k \in [\![2, p-1]\!]$
3: **for** $i = 1$ jusqu'à $n-1$ **do**
4: $\quad C_{2,i} := d_{1,i}$
5: **end for**
6: **for** $k = 3$ jusqu'à $p-1$ **do**
7: $\quad$ **for** $i = k$ jusqu'à $n-1$ **do**
8: $\quad\quad C_{k,i} := \max_{j \in [\![k-1, i-1]\!]}(C_{k-1,j} + d_{j,i})$
9: $\quad$ **end for**
10: **end for**
11: $j := \arg\max_{j \in [\![p-1, n-1]\!]}(C_{p-1,j} + d_{j,n})$
12: $OPT := C_{p-1,j} + d_{j,n}$
13: Initialiser $i := j$ et $\mathcal{J} := \{1, j, n\}$
14: **for** $k = p-1$ jusqu'à 3 avec décrément $k := k-1$ **do**
15: $\quad j := \arg\max_{j' \in [\![k-1, i-1]\!]}(C_{k-1,j'} + d_{j',i})$
16: $\quad$ Ajouter j à $\mathcal{J}$
17: $\quad i := j$
18: **end for**
19: **return** OPT : le coût optimal, et l'ensemble des indices sélectionnés $\mathcal{J}$

3.1.2 Algorithme de dispersion Max-Min (p-dispersion-Mm)

Nous considérons ici le problème de sélection de p points dans un front de Pareto 2D, de sorte à maximiser la distance minimale entre deux points consécutifs dans la solution. Ce critère est appelé *Max-Min p-dispersion*. L'objectif est donc de trouver un sous-ensemble ordonné $S = \{x_{i_1}, x_{i_2}, \ldots, x_{i_p}\}$ tel que :

$$\mathrm{OPT}_{\mathrm{Mm}} = \max_{S \subseteq E, |S|=p} \left(\min_{1 \leq j < p} d(x_{i_j}, x_{i_{j+1}}) \right)$$

L'algorithme repose sur une approche dynamique classique, adaptée au cadre du front de Pareto 2D trié (selon le Lemme 1). Il s'appuie sur une structure $C^{\mathrm{Mm}}_{k,i}$ représentant, pour chaque $k \in [\![2, p-1]\!]$ et chaque point i, la meilleure valeur possible du critère pour une sélection de $k+1$ points se terminant en x_i. À chaque étape, la mise à jour repose sur le schéma suivant :

$$C^{\mathrm{Mm}}_{k,i} = \max_{j \in [\![k-1, i-1]\!]} \min(C^{\mathrm{Mm}}_{k-1,j}, d_{j,i})$$

Pour cela, on utilise la procédure suivante :

Algorithm 2 Calcul de $\max_{j\in[\![k-1,i-1]\!]} \min(C^{\mathrm{Mm}}_{k-1,j}, d_{j,i})$

Définir $a := k-1$, $b := i$
while $b - a \geq 2$ **do**
    Calculer $j = \left\lfloor \frac{a+b}{2} \right\rfloor$
    if $C^{\mathrm{Mm}}_{k-1,j} - d_{j,i} > 0$ **then**
        $b := j$
    else
        $a := j$
    end if
end while
return $\max\left(\min(C^{\mathrm{Mm}}_{k-1,a}, d_{a,i}), \min(C^{\mathrm{Mm}}_{k-1,b}, d_{b,i})\right)$

Le cœur de l'algorithme dynamique s'appuie alors sur les étapes suivantes :

Algorithm 3 Max-min p-dispersion dans un front de Pareto 2D avec $p \geq 3$

Entrée : n points d'un front de Pareto 2D, $E = \{x_i\}_{i\in[\![1,n]\!]}$; un entier $p \in [\![3, n]\!]$.
Reindexer E selon l'ordre du Lemme 1
Initialiser un vecteur C avec $C_i := d_{1,i}$ pour $i \in [\![2, n-1]\!]$
for $k = 3$ jusqu'à $p-1$ avec incrément $k \leftarrow k+1$ **do**
    for $i = n-1$ jusqu'à k avec décrément $i \leftarrow i-1$ **do**
        $C_i := \max_{j\in[\![k-1,i-1]\!]} \min(C_j, d_{j,i})$ avec l'Algorithme 1
    end for
end for
$OPT := \max_{j\in[\![p,n-1]\!]} \min(C_j, d_{j,n})$ avec l'Algorithme 1
return OPT et une solution obtenue avec l'Algorithme 2 (ou 3)

Une fois le coût optimal calculé, il est possible de reconstruire la solution associée à l'aide d'un algorithme de backtracking. Deux variantes sont possibles selon que l'on parcourt les indices dans l'ordre croissant ou décroissant :

Algorithm 4 Algorithme de backtracking utilisant $O(n)$ d'espace (croissant)

Initialiser $M := 1$, $m := 1$, $S = \{1, n\}$.
for $k = 2$ jusqu'à $p-1$ avec incrément $k \leftarrow k+1$ **do**
    $M :=$ le plus petit indice tel que $d(x_m, x_M) \geq OPT$
    Ajouter M à S
    $m := M$
end for
return S

Ces algorithmes garantissent, pour un p donné, une solution optimale en temps polynomial grâce à la structure du front de Pareto 2D et l'usage d'une structure dynamique compacte.

Algorithm 5 Algorithme de backtracking utilisant $O(n)$ d'espace (décroissant)

1: Initialiser $M := n$, $m := n$, $S = \{1, n\}$.
2: **for** $k = p - 1$ jusqu'à 2 avec décrément $k \leftarrow k - 1$ **do**
3: $m :=$ le plus grand indice tel que $d(x_m, x_M) \geq OPT$
4: Ajouter m à S
5: $M := m$
6: **end for**
7: **return** S

3.1.3 Algorithme dynamique pour le problème *p*-dispersion-MSm

La variante Max-Sum-Min (MSm) vise à maximiser la somme des minimums entre distances successives dans une sous-séquence ordonnée de p points du front de Pareto. Plus formellement, pour une séquence ordonnée $(x_1, x_2, \ldots, x_p)$, on cherche à maximiser :

$$\sum_{i=2}^{p-1} \min\left(d(x_{i-1}, x_i), d(x_i, x_{i+1})\right)$$

Cette variante est particulièrement pertinente dans les situations où l'on souhaite garantir à la fois une dispersion globale suffisante tout en évitant la présence de points trop isolés dans le sous-ensemble sélectionné.

Une approche efficace pour résoudre ce problème repose sur la programmation dynamique. L'algorithme construit récursivement les meilleures solutions pour des sous-fronts de taille croissante, en maintenant pour chaque triplet de positions la meilleure valeur atteignable. Il repose sur une structure tridimensionnelle, avec une complexité en temps de $\mathcal{O}(pn^3)$ et en espace de $\mathcal{O}(pn^2)$.

Algorithm 6 Dispersion Max-Somme-Min pour $p > 5$ dans un front de Pareto 2D

Entrée : $E = \{x_i\}_{i \in [\![1,n]\!]}$ un front de Pareto 2D de taille n ; un entier $p \in [\![6, n]\!]$.
Réindexer E selon l'ordre du Lemme 1.
Initialiser une matrice C avec $C_{k,i',i''} := 0$ pour tout $k \in [\![2, p-1]\!]$, $(i', i'') \in [\![1, n]\!]^2$.
for $i' = 3$ jusqu'à n **do**
    for $i'' = 2$ jusqu'à $i' - 1$ **do**
        $C_{3,i',i''} := d_{1,i''} + \min(d_{1,i'}, d_{i'',i'})$
    end for
end for
for $k = 4$ jusqu'à p **do**
    for $i' = k$ jusqu'à n **do**
        for $i'' = k - 1$ jusqu'à $i' - 1$ **do**
            $C_{k,i',i''} := \max_{j \in [\![k-2, i''-1]\!]} (C_{k-1,j,i''} + \min(d_{j,i'}, d_{i'',i'}))$
        end for
    end for
end for
$j := \text{argmax}_{j' \in [\![p-2, n-1]\!]} (C_{p,j',n} + d_{j',n})$
$OPT := C_{p,j,n} + d_{j,n}$
Initialiser $i' := n$, $i'' := j$ et $S := \{1, j, n\}$
for $k = p - 1$ jusqu'à 3 avec décrément $k \leftarrow k - 1$ **do**
    Calculer $j := \text{argmax}_{j' \in [\![k-2, i''-1]\!]} (C_{k-1,j',i''} + \min(d_{j',i'}, d_{i'',i'}))$
    Ajouter j à S, puis mettre à jour $i' := i''$ et $i'' := j$
end for
Retourner OPT le coût optimal, ainsi que l'ensemble d'indices sélectionnés S

3.1.4 Hypervolume pondéré (WHV)

Pour refléter les **préférences utilisateur**, une version pondérée de l'hypervolume [5] peut être utilisée, où la région dominée est intégrée selon une densité $w(x)$, souvent choisie comme une fonction $\rho(x)$ définissant les zones préférées :

$$WHV(S) = \int_{\mathcal{R}(S)} w(x)\, dx \quad \text{où} \quad w(x) = \rho(x)$$

Cette version pondérée permet de favoriser certaines zones de l'espace objectif, par exemple en concentrant la sélection sur des régions jugées plus pertinentes selon les priorités du décideur. Elle est particulièrement utile dans un contexte d'aide multicritère à la décision.

Le problème de la sélection de sous-ensemble maximisant l'hypervolume pondéré (HSSP, pour *Hypervolume Subset Selection Problem*) est difficile en général, mais une solution exacte est possible en dimension 2. Dans ce cas, les points peuvent être triés selon l'ordre du premier objectif, et une approche dynamique ascendante permet d'itérer efficacement sur les sous-ensembles de taille croissante.

Plus précisément, l'algorithme proposé par Auger et al. [5] exploite la structure du front de Pareto 2D et la propriété d'additivité de la contribution marginale de chaque point pour construire une solution optimale. Cette approche repose sur une fonction

$C_H^w(\vec{l}, \vec{u})$ qui évalue l'hypervolume pondéré d'un rectangle défini par deux coins opposés, ce qui permet d'agréger efficacement les contributions.

Algorithm 7 Solveur HSSP(O, q, r, C_H^w)

Require: Une matrice $O := (o_{i,j})_{p\times 2}$, où les lignes représentent les vecteurs objectifs triés par ordre croissant selon le premier objectif ; une taille de sous-ensemble $q \geq 1$; un point de référence $r = (r_1, r_2)$.

1: $C_H^w(\vec{l}, \vec{u})$ retourne l'hypervolume pondéré du rectangle défini entre $\vec{l}$ et $\vec{u}$.

Ensure: Renvoie un ensemble S de références aux lignes de O qui maximise l'hypervolume pondéré.

2: **for** $1 \leq c \leq p$ **do**
3: $\quad h_1^c \leftarrow h_c := C_H^w(\{(o_{1,1}, o_{1,2})\}, r)$
4: $\quad S_1^c \leftarrow \{c\}$ ▷ sous-ensembles optimaux
5: **end for**
6: **for** $t = 2$ à q **do** ▷ approche ascendante (bottom-up)
7: $\quad$ **for** $c = q - t + 1$ à $p - t + 1$ **do**
8: $\quad\quad l \leftarrow (0, \ldots, 0)$
9: $\quad\quad$ **for** $d = c + 1$ à $p - t + 1$ **do**
10: $\quad\quad\quad l_d \leftarrow h_{t-1}^d + C_H^w(\{(o_{c,1}, o_{c,2})\}, (o_{d,1}, r_2))$
11: $\quad\quad$ **end for**
12: $\quad\quad m \leftarrow \arg\max_d l_d$
13: $\quad\quad S_t^c \leftarrow \{c\} \cup S_{t-1}^m$ ▷ fusionne c avec S_m et ...
14: $\quad\quad h_t^c \leftarrow l_m$ ▷ ... met à jour l'hypervolume
15: $\quad$ **end for**
16: **end for**
17: $m \leftarrow \arg\max_i h_q^i$ ▷ choisir le meilleur sous-ensemble
18: **return** S_q^m ▷ solution au problème global

4 Analyse comparative des critères de sélection

4.1 Conditions expérimentales

Les tests ont été effectués sur des instances de fronts de Pareto issues du dépôt public suivant :

`https://github.com/Tiralex1/solverCO`

Ces instances ont été générées dans le cadre de l'étude [9] : Elles sont directement accessibles et permettent de reproduire les expériences décrites dans cette section.

Pour l'évaluation expérimentale, les tests sur les petites instances ont été réalisés sur une machine personnelle avec les caractéristiques suivantes :

- **Processeur** : Intel Core i5-1335U, 10 cœurs (2 Performance + 8 Efficient), fréquence maximale 4.6 GHz.
- **Mémoire vive (RAM)** : 31 Go.
- **Système d'exploitation** : Debian GNU/Linux 12 (Bookworm), noyau Linux 6.12.22.

4.2 Premier test : nombre de solutions optimales

Ce test a pour objectif de mesurer le **nombre de solutions optimales** obtenues par les différents critères de sélection pour des instances simples ($n = 50$) et différentes valeurs de $a \in \{0.5, 1, 2\}$.

Table 2: Nombre moyen de solutions optimales selon le critère (n = 50, a = 1)

p	HSS	Mm	MSN	MSm	MmS	MS
3	1.00	6.60	1.00	1.00	1.00	1.00
4	1.00	24.70	1.00	1.00	1.00	1.00
5	1.00	331.20	5.50	3.60	1.00	1.00

Table 3: Nombre moyen de solutions optimales selon le critère (n = 50, a = 2)

p	HSS	Mm	MSN	MSm	MmS	MS
3	1.00	6.60	1.00	1.00	1.00	1.00
4	1.00	24.70	1.00	1.00	1.00	1.00
5	1.00	331.20	1.00	1.00	1.00	1.00

Table 4: Nombre moyen de solutions optimales selon le critère (n = 50, a = 0.5)

p	HSS	Mm	MSN	MSm	MmS	MS
3	1.00	6.60	1.00	1.00	1.00	1.00
4	1.00	24.70	1.00	1.00	1.00	1.00
5	1.00	331.20	1.00	1.00	1.00	1.00

On observe que le critère **Max-Min (Mm)** génère un nombre extrêmement élevé de solutions optimales, et ce de manière systématique, indépendamment de la valeur du paramètre a. Cette invariance vis-à-vis de a s'explique par la nature même du critère Max-Min, qui ne dépend que de la distance minimale entre les points sélectionnés, sans considération de pondération ou d'agrégation non linéaire. Ainsi, tant que plusieurs sous-ensembles atteignent la même valeur maximale de distance minimale, ils sont tous considérés comme optimaux. Cette situation est typique dans des instances où la distribution des points offre de nombreuses configurations équivalentes du point de vue de la distance minimale. Cela suggère une forte **dégénérescence du critère Max-Min**, et donc une faible capacité à discriminer les solutions dans des fronts de Pareto denses ou réguliers.

En revanche, les autres critères (HSS, MSm, MmS, MS) produisent presque systématiquement une unique solution optimale, quelle que soit la valeur de p ou de a. Cela indique qu'ils sont plus **discriminants**, probablement en raison de leur nature multi-objectif ou de leur structure non linéaire (somme, minimum sur des sommes, etc.), qui rend plus rare l'égalité parfaite de score entre plusieurs solutions.

Une exception partielle est toutefois observée pour les critères **Max-Sum-Min (MSm)** et **Max-Sum-Neighbor (MSN)** lorsque $p = 5$ et $a = 0.5$, où le nombre moyen de solutions optimales augmente significativement (5.5 pour MSN, 3.6 pour MSm). Cela peut s'expliquer par la valeur faible de a, qui réduit l'impact des grandes distances dans la fonction d'agrégation (c'est-à-dire que la somme des distances devient moins sensible aux grandes valeurs). Ainsi, plusieurs solutions peuvent aboutir à des scores similaires voire égaux, entraînant une forme de dégénérescence comparable (bien que moins prononcée) à celle du critère Max-Min.

Cependant, ce phénomène ne se manifeste pas pour $a = 1$ et $a = 2$, ce qui suggère que la **valeur de** a joue un rôle important dans la capacité de certains critères à distinguer finement les solutions. En particulier, lorsque $a > 1$, les grandes distances sont amplifiées, ce qui favorise la discrimination entre les sous-ensembles et limite le risque d'ex æquo.

Cette analyse met en évidence que le choix du critère, ainsi que celui du paramètre a, influence fortement la structure de l'espace des solutions optimales. Les critères peu discriminants comme Max-Min peuvent conduire à de nombreuses solutions équivalentes, ce qui complique la sélection finale. À l'inverse, des critères plus sensibles (comme MSm, MSN ou MS) permettent une sélection plus fine, surtout pour des valeurs de a modérées ou élevées.

4.3 Second test : compatibilité entre critères (ratios de dispersion)

Ce test vise à **évaluer la compatibilité entre les critères de sélection**, en analysant dans quelle mesure une solution optimale selon un critère donné reste performante pour les autres critères. Il repose sur une **évaluation croisée des performances** : pour chaque solution optimale produite par un critère A, on calcule son coût selon tous les autres critères B, puis on le rapporte au coût de la solution optimale selon B lui-même. Le résultat est un **ratio de dispersion**, supérieur ou égal à 1, qui mesure la dégradation de performance d'une solution lorsqu'elle est évaluée avec un critère différent de celui qui l'a générée.

Cette approche permet de mieux comprendre les **relations de compatibilité ou d'incompatibilité entre critères** : des ratios proches de 1 indiquent une bonne compat-

ibilité, tandis que des ratios élevés signalent une divergence importante entre les objectifs des critères comparés.

Ce protocole de comparaison s'inspire d'un cadre méthodologique proposé dans [7], appliqué ici à notre contexte spécifique de sélection de points dans un front de Pareto avec des critères de dispersion.

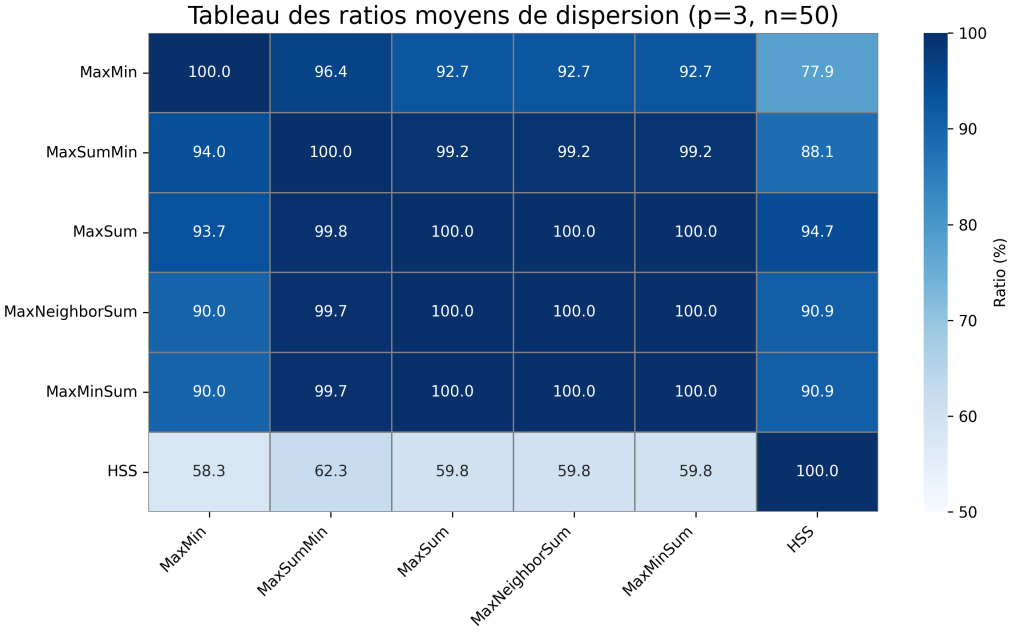


Figure 5: Ratios de dispersion (p=3, n=50)

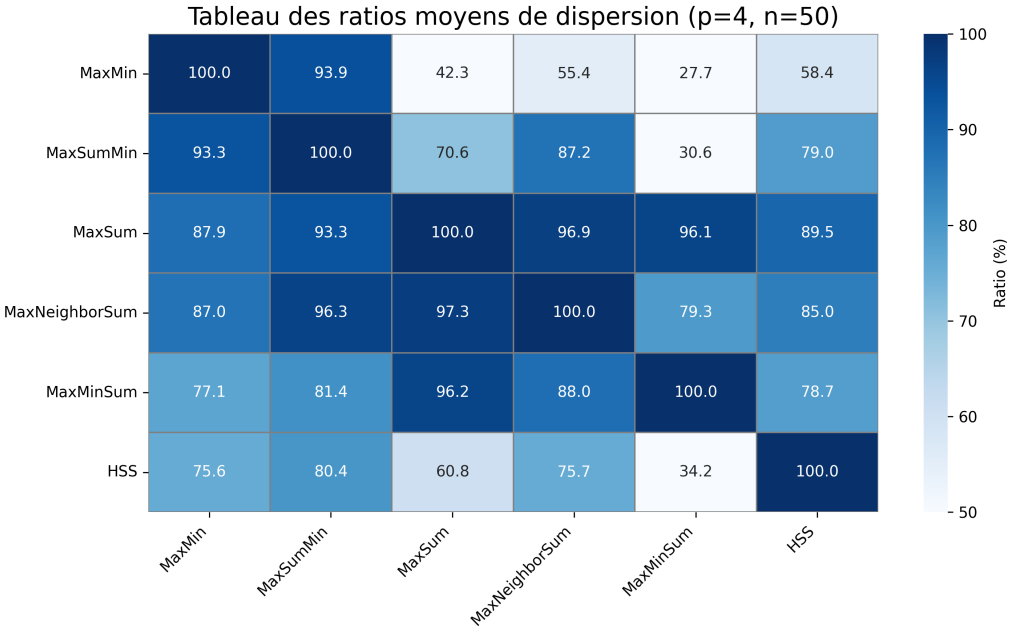


Figure 6: Ratios de dispersion (p=4, n=50)

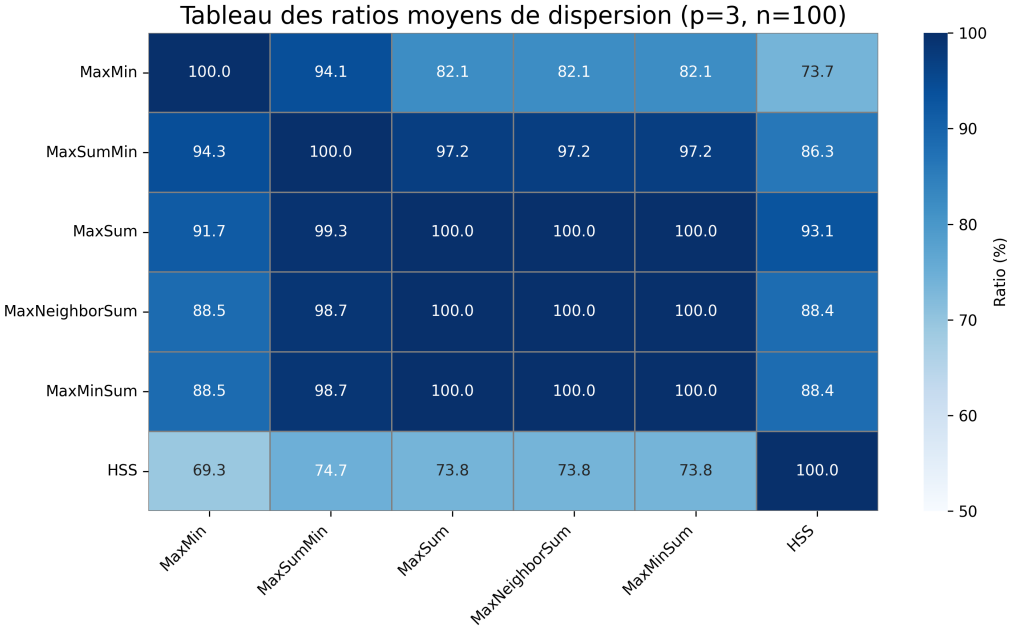


Figure 7: Ratios de dispersion (p=3, n=100)

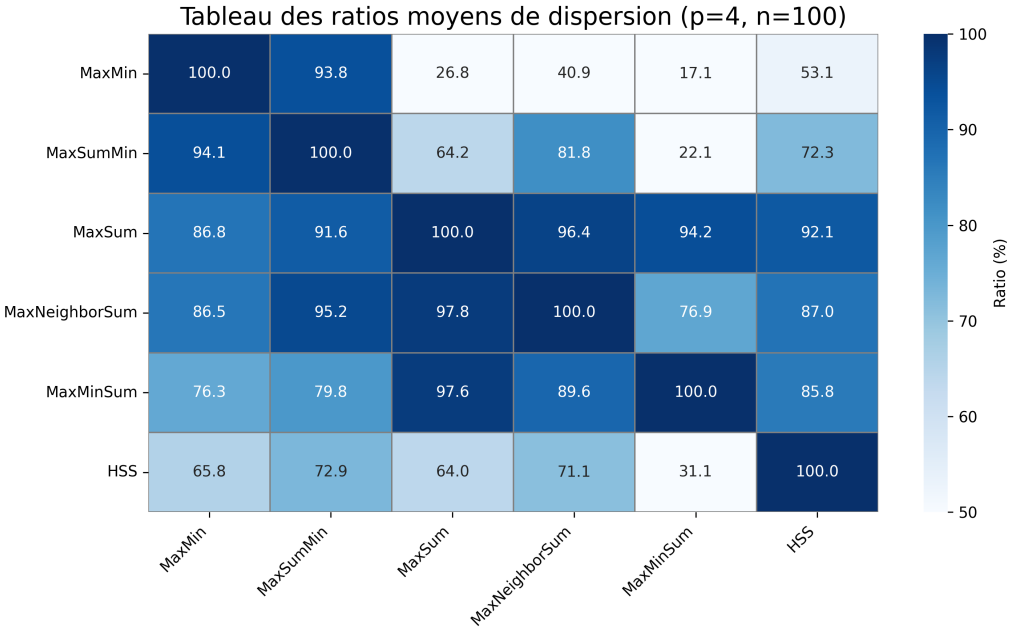


Figure 8: Ratios de dispersion (p=4, n=100)

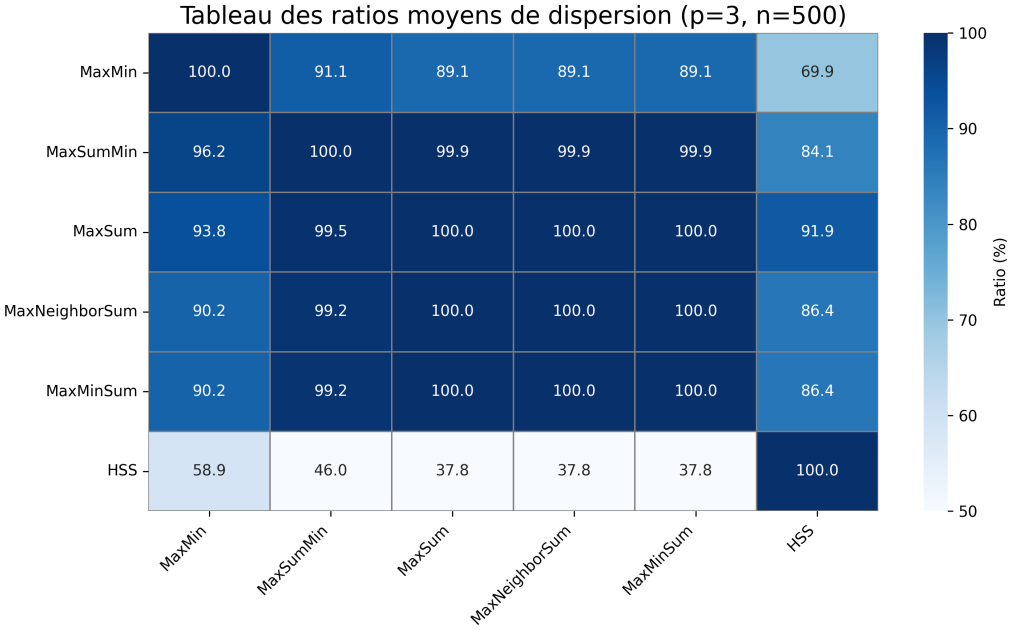


Figure 9: Ratios de dispersion (p=3, n=500)

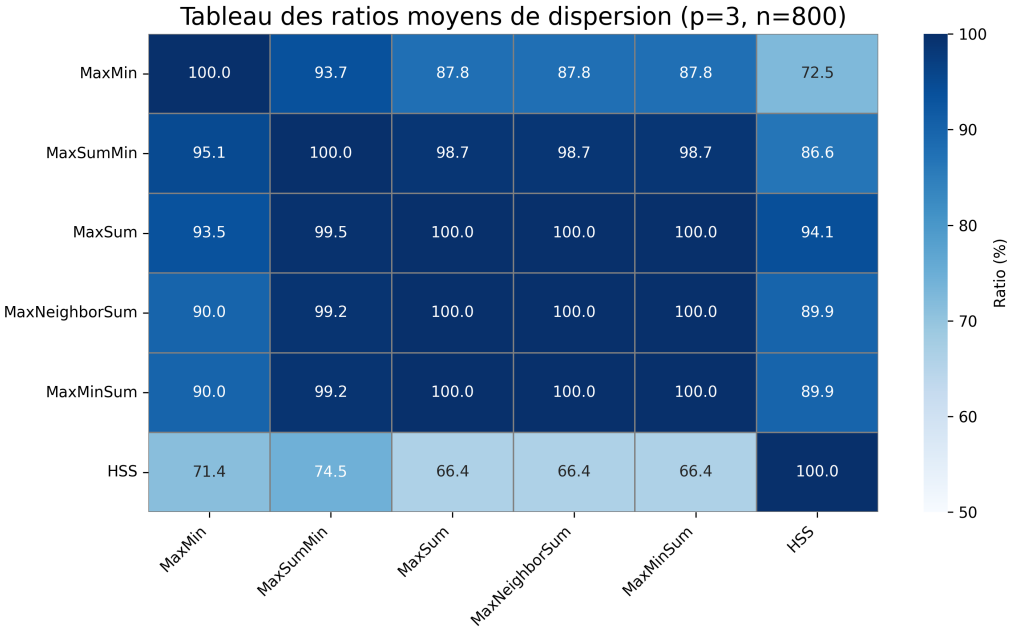


Figure 10: Ratios de dispersion (p=3, n=800)

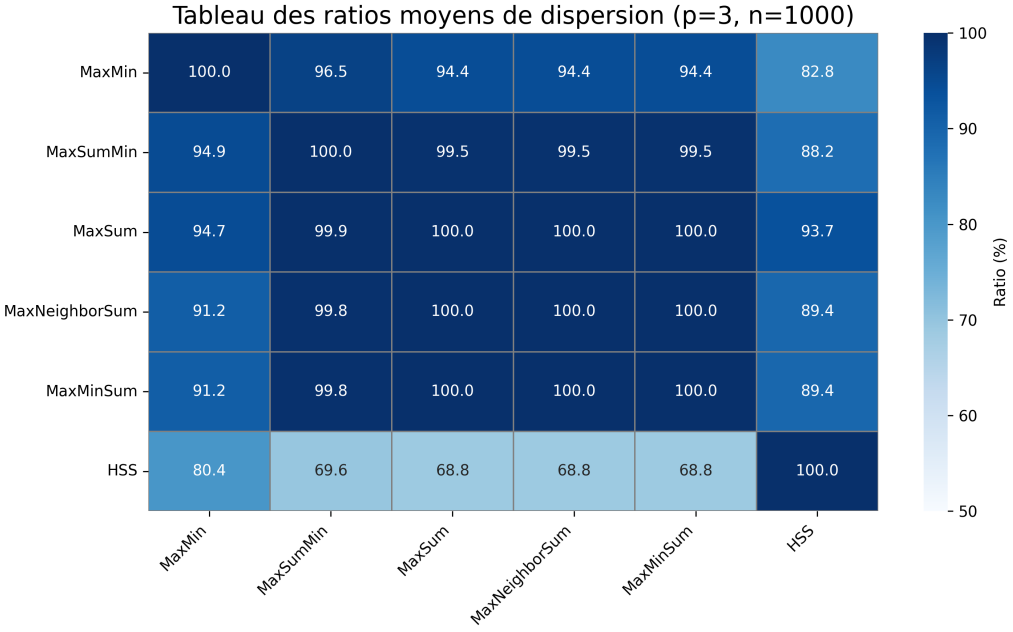


Figure 11: Ratios de dispersion (p=3, n=1000)

Conclusion. Les résultats précédents permettent de tirer plusieurs constats importants :

- Pour $p = 3$, on observe une forte compatibilité entre les solveurs *MaxSum* (MS), *MaxSumMin* (MSm), *MaxNeighborSum* (MSN) et *MaxMinSum* (MmS). Les ratios croisés entre ces méthodes dépassent souvent 99%, indiquant qu'elles produisent des solutions très similaires, quel que soit le critère d'évaluation utilisé. Cette compatibilité reste stable quel que soit le nombre de points dans le front (n), ce qui suggère une robustesse structurelle dans leurs comportements pour une faible cardinalité p.

- En revanche, lorsque $p = 4$, une baisse notable de compatibilité apparaît. En particulier, les solutions produites par *MaxSum*, *MaxSumMin* et *MaxNeighborSum* deviennent beaucoup moins efficaces lorsqu'elles sont évaluées selon le critère *MaxMin*. Cela indique une divergence plus marquée entre les objectifs de dispersion minimale et ceux favorisant la somme ou l'agrégation des distances. Ce phénomène s'explique probablement par le fait que, lorsque p augmente, le compromis entre maximiser les distances minimales et maximiser la dispersion globale devient plus difficile à atteindre.

- Dans tous les cas, le solveur *HSS* montre une compatibilité significativement plus faible avec les autres méthodes. Même pour $p = 3$, ses performances selon les critères de dispersion sont inférieures. Cette différence s'explique par le fait que *HSS* optimise une mesure purement géométrique — l'hypervolume — qui ne prend pas en compte les distances inter-points. Par conséquent, même si la couverture du front est efficace du point de vue de l'hypervolume, cela n'implique pas une bonne répartition spatiale des solutions.

Ces observations montrent que :

- Les critères fondés sur la somme ou l'agrégation des distances (MS, MSm, MSN, MmS) tendent à être compatibles entre eux pour de petites valeurs de p, indépendamment de la taille de l'ensemble.
- L'incompatibilité avec *MaxMin* s'accentue à mesure que p augmente, ce qui suggère une tension croissante entre dispersion locale (distance minimale) et dispersion globale (somme des distances).
- Le comportement isolé de *HSS* souligne que l'optimisation de l'hypervolume suit une logique différente, et ne peut être directement assimilée à une stratégie de dispersion selon les distances.

4.4 Second test : évaluation des performances du critère MSN

Cette seconde partie du test adopte une démarche similaire à celle de la première partie : elle repose sur l'évaluation croisée des solutions obtenues par un critère donné en utilisant les fonctions de coût des autres critères. Toutefois, ici, l'analyse se concentre uniquement sur les solutions générées par le **critère MaxNeighborSum (MSN)**, pour différentes valeurs du paramètre a. L'objectif est de mieux comprendre la **stabilité et la robustesse** de MSN face aux variations de ce paramètre, en observant dans quelle mesure les solutions qu'il propose restent performantes lorsqu'elles sont évaluées selon les autres critères. Cela permet également de comparer plus finement le comportement de MSN aux approches alternatives dans des contextes variés.

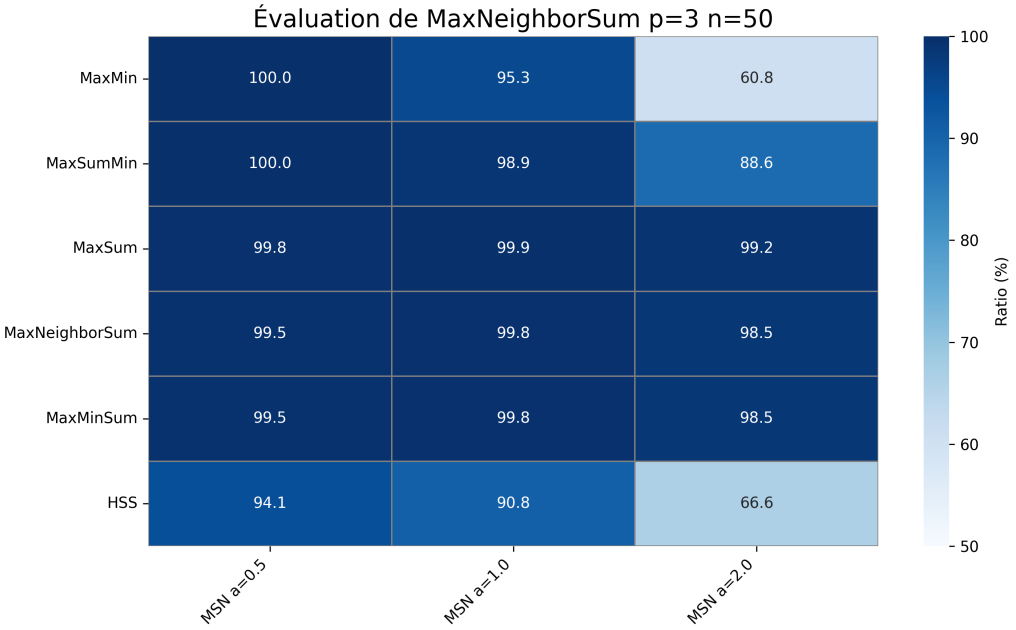


Figure 12: Évaluation MSN (p=3, n=50)

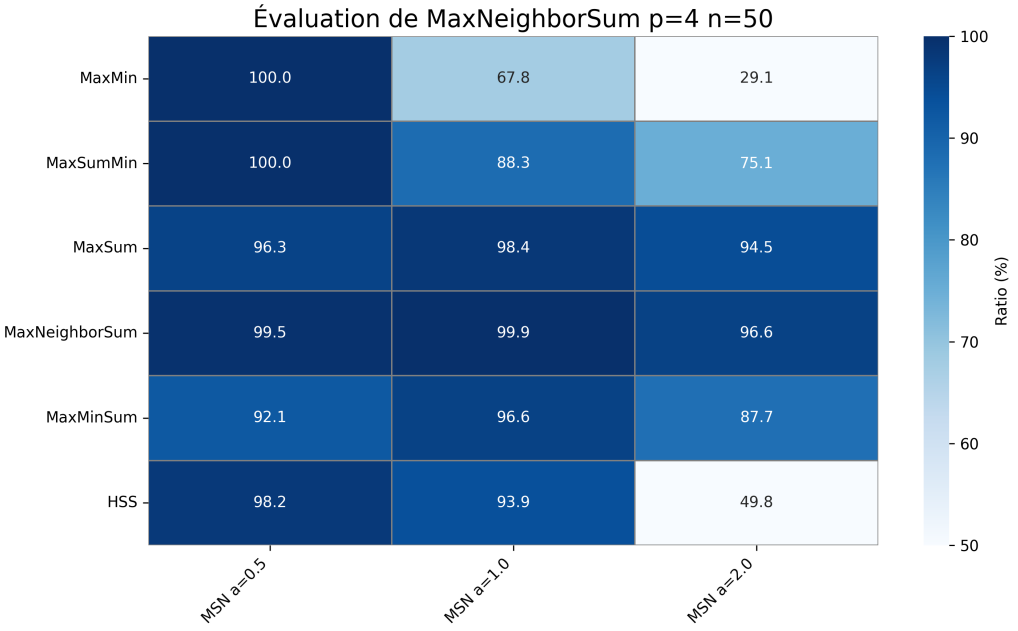


Figure 13: Évaluation MSN (p=4, n=50)

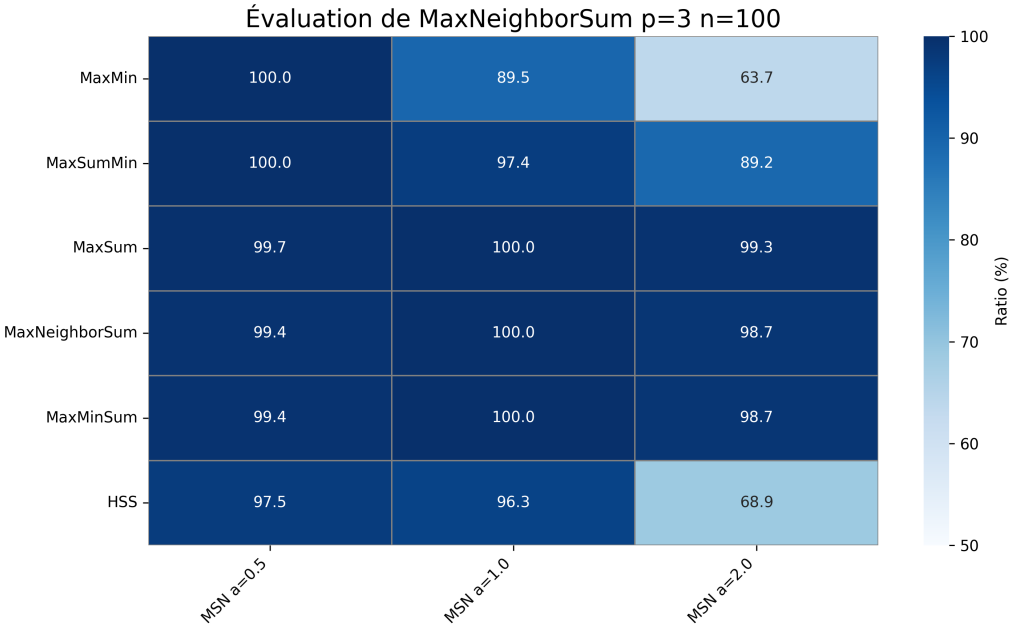


Figure 14: Évaluation MSN (p=3, n=100)

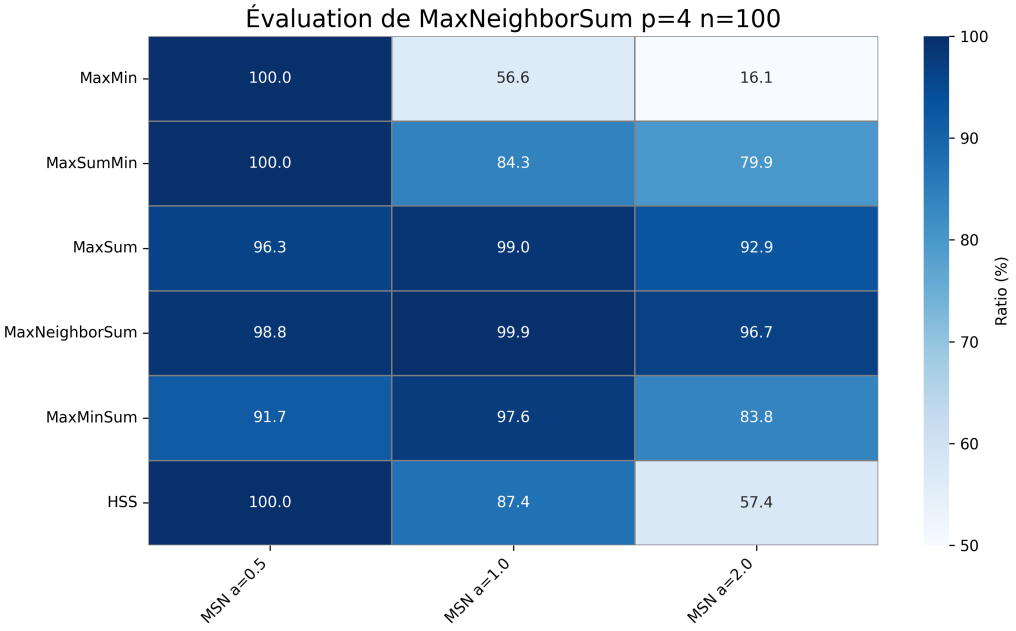


Figure 15: Évaluation MSN (p=4, n=100)

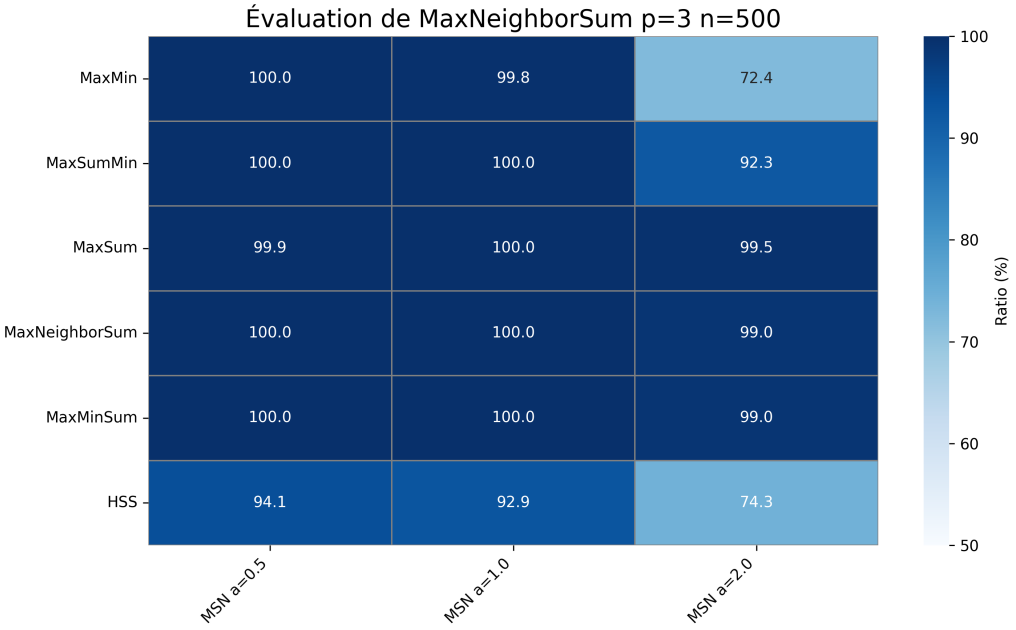


Figure 16: Évaluation MSN (p=3, n=500)

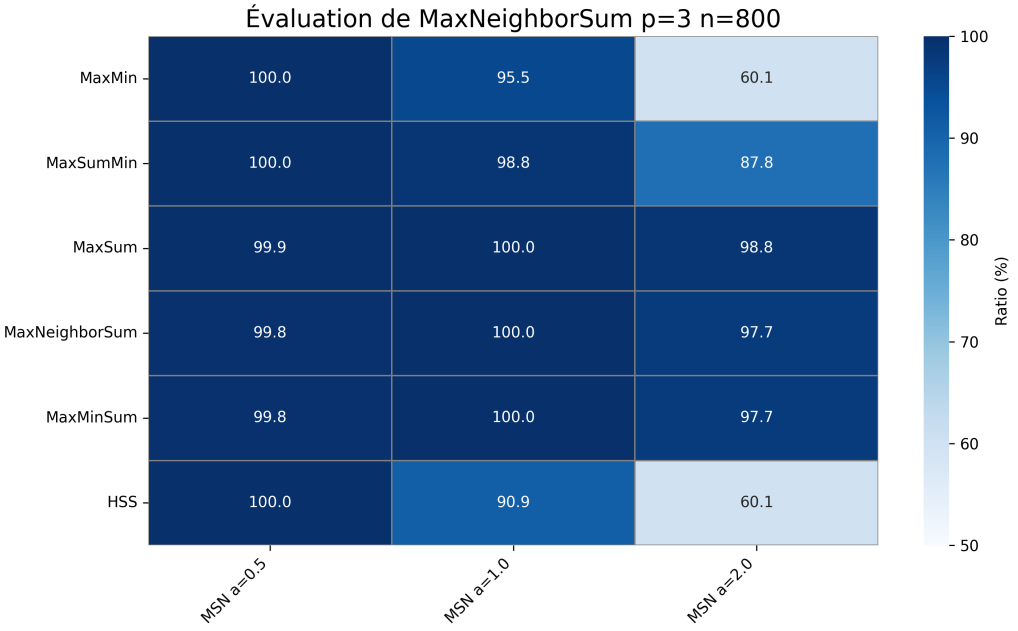


Figure 17: Évaluation MSN (p=3, n=800)

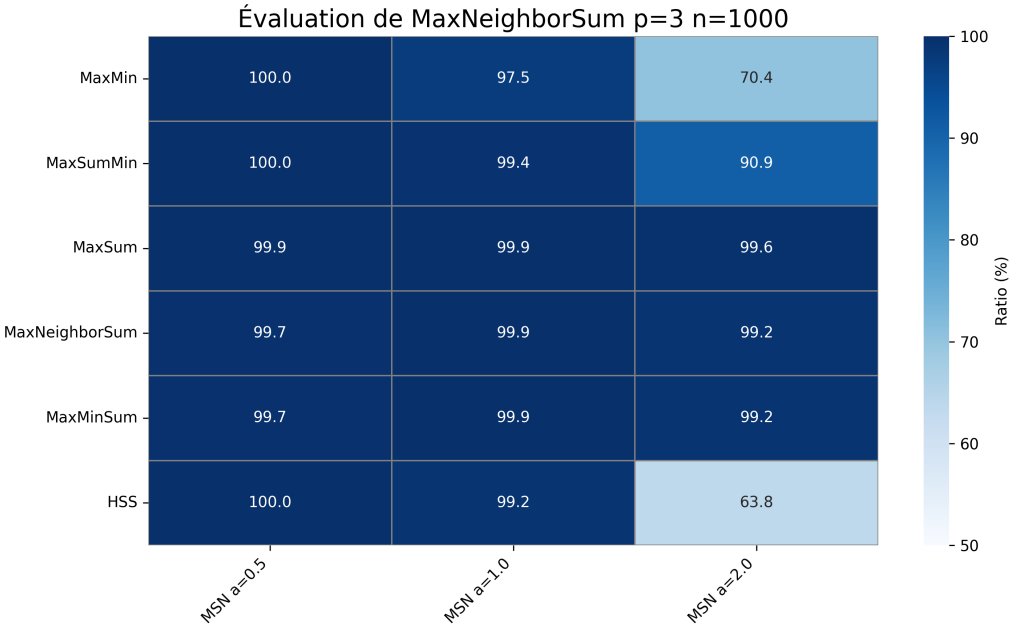


Figure 18: Évaluation MSN (p=3, n=1000)

Conclusion.

L'analyse des performances des solutions issues du critère *MaxNeighborSum* (MSN) montre une influence notable du paramètre α sur leur compatibilité avec les autres critères. En particulier :

- Pour les petites valeurs de α (notamment $\alpha = 0.5$), les solutions MSN présentent une très forte compatibilité avec l'ensemble des critères testés, dépassant souvent 99%.

Cela indique une grande stabilité des solutions MSN dans ce régime, qui restent performantes quelle que soit la fonction de coût utilisée pour leur évaluation.

- Lorsque α augmente (par exemple $\alpha = 2.0$), la compatibilité décroît sensiblement, en particulier vis-à-vis des critères *MaxMin* et *HSS*. Cette baisse suggère que les solutions MSN deviennent plus spécialisées et moins robustes face à des mesures de dispersion plus locales ou géométriques.

- De manière générale, MSN se distingue par une bonne polyvalence pour des valeurs modérées à faibles de α, ce qui en fait un critère robuste dans des contextes variés.

Ces observations confirment que le choix de α est un paramètre clé dans la modélisation du critère MSN, impactant directement la nature et la qualité des solutions obtenues par rapport aux autres approches de dispersion.

5 Architecture du programme et choix de conception

L'implémentation du projet repose sur une architecture orientée objet modulaire, facilitant l'extension à de nouveaux solveurs ou à de nouveaux critères de sélection. Le cœur du système est constitué d'une hiérarchie de classes permettant de factoriser les traitements communs tout en spécialisant les comportements pour chaque méthode de sélection.

5.1 Classe de base `PF_solver`

La classe `PF_solver` hérite de la classe `SolverCO` et constitue le cadre de tout solveur opérant sur un front de Pareto 2D. Elle encapsule les éléments communs à toutes les variantes du problème, notamment :

- les données d'entrée : ensemble de points (`Point_set`), nombre de points à sélectionner (K) ;
- les structures de sortie : solution courante, coût associé ;
- des méthodes utilitaires : chargement des données, vérification de la validité de la solution, etc.

Cette classe fournit donc tous les éléments de base nécessaires pour la résolution d'un problème d'optimisation sur un front de Pareto.

5.2 Classe spécialisée `PFSelectionSolver`

La classe `PFSelectionSolver` hérite de `PF_solver` et constitue un cadre dédié à la sélection de sous-ensembles de p points. Elle introduit des fonctionnalités supplémentaires spécifiques aux méthodes de sélection, notamment :

- le type de distance à utiliser (Minkowski ou Chebyshev) ;
- le paramètre α d'agrégation pour les critères pondérés ;
- des structures pour stocker les indices sélectionnés par différentes approches ;
- des fonctions d'évaluation de la qualité de la sélection selon plusieurs critères : dispersion (max-min, max-sum, etc.) ou hypervolume (HSS) ;
- la méthode abstraite `solve()` à redéfinir dans chaque solveur spécifique.

Cette classe centralise donc l'ensemble des outils nécessaires à la mise en œuvre et à la comparaison de stratégies de sélection multi-objectifs.

5.3 Classes dérivées spécialisées

Chaque solveur est implémenté dans une classe dérivée de `PFSelectionSolver`, et redéfinit la méthode `solve()` en fonction du critère d'optimisation visé. Ces solveurs incluent :

- `PFMaxMinSolver`

- `PFMaxSumSolver`
- `PFMaxMinSumSolver`
- `PFMaxSumMinSolver`
- `PFMaxNeighborSumSolver`
- `PFHypervolumeSolver`

Chaque solveur implémente des variantes exactes basées sur un algorithme naïf pour $p = 3$, 4, ou 5 en testant toutes les combinaisons possibles. De plus, pour les critères `max-min` et `MSN`, des algorithmes efficaces de programmation dynamique ont été intégrés, ce qui permet d'obtenir des résultats exacts sur des instances de taille plus importante.

5.4 Choix techniques et extensibilité

La conception du programme repose sur les principes suivants :

- **Héritage structuré** : les classes dérivent logiquement les unes des autres, permettant une factorisation maximale du code commun.
- **Encapsulation des données** : les structures de données (points, solution, paramètres) sont protégées et manipulées via des accesseurs.
- **Modularité forte** : chaque solveur est développé dans un fichier séparé, facilitant les ajouts et modifications.
- **Comparabilité** : toutes les méthodes utilisent une base commune d'évaluation et peuvent donc être comparées équitablement.

5.5 Environnement global du projet

Le projet s'intègre dans une architecture logicielle unifiée centrée autour du gestionnaire principal `SolverCO`, qui regroupe plusieurs solveurs dédiés à des problèmes d'optimisation combinatoire variés. Chaque problème est représenté par une classe dédiée, assurant une séparation claire des responsabilités.

La branche `PF_solver` se subdivise en deux modules spécialisés :

- `PFSelectionSolver` pour la sélection de sous-ensembles de p points selon divers critères (dispersion, HSS) ;

- `Solver`$_{c}lusterpourleregroupementdepointsenclusters$.

  Cette structure favorise :

  - la mutualisation des outils (lecture, distances, métriques) ;
  - le débogage modulaire et localisé ;
  - la réalisation de comparaisons systématiques entre critères et méthodes dans un environnement cohérent et extensible.

6 Conclusions et perspectives

Au cours de ce stage, nous avons étudié plusieurs méthodes exactes et heuristiques pour résoudre des variantes du problème de sélection de p points dans un front de Pareto 2D, en nous concentrant principalement sur des critères de dispersion tels que Max-Min, Max-Sum, Max-Min-Sum, Max-Sum-Min et Max-Neighbor-Sum. L'objectif était de maximiser la diversité spatiale des points sélectionnés en fonction de différentes métriques de distance, avec la possibilité d'intégrer un paramètre d'agrégation α permettant d'ajuster la sensibilité du critère choisi.

Les différents solveurs implémentés, s'appuyant sur une architecture orientée objet modulaire, ont permis de comparer les performances des variantes proposées de manière systématique, à la fois en termes de qualité de solution et de comportement croisé entre critères.

L'analyse des résultats a montré que chaque critère induit un comportement de sélection spécifique, avec des compromis différents entre dispersion locale et globale.

Perspectives et publication

Ce travail constitue une base solide pour une éventuelle publication scientifique. Cependant, plusieurs points restent à approfondir, notamment la comparaison plus approfondie avec des méthodes de clustering classiques, qui représentent une approche alternative pour sélectionner un sous-ensemble représentatif. Cette comparaison pourrait mettre en lumière les avantages et limites respectifs des méthodes de sélection et de clustering dans le contexte de fronts de Pareto.

Comparaison sélection vs. clustering

La sélection de points par critères de dispersion, telle qu'explorée dans ce stage, vise à maximiser la diversité tout en garantissant une bonne couverture du front. En revanche, les méthodes de clustering regroupent les points selon leur proximité, ce qui peut conduire à des sous-ensembles plus homogènes mais potentiellement moins dispersés. Une étude comparative approfondie permettrait de mieux comprendre les situations où l'une ou l'autre approche est préférable, en fonction des objectifs spécifiques d'analyse multi-objectif.

Ce stage m'a permis d'approfondir mes compétences en C++, en conception logicielle orientée objet et en programmation algorithmique. J'ai également acquis une meilleure compréhension des problèmes liés à l'optimisation combinatoire. J'espère à l'avenir continuer à explorer ces thématiques dans le cadre de projets de recherche ou industriels.

References

[1] N. Dupin. Polynomial algorithms for p-dispersion problems in a planar Pareto Front. *RAIRO-Operations Research*, 57(2):857–880, 2023.

[2] N. Dupin, F. Nielsen, E. Talbi.
Unified Polynomial Dynamic Programming Algorithms for p-Center Variants in a 2D Pareto Front.
arXiv:2002.11830, 2020.
`https://arxiv.org/pdf/2002.11830`

[3] Qing Cai, Licheng Jiao, Feng Gao, Xinwen Hou, Shuang Yang.
A survey on network community detection based on evolutionary computation.
International Journal of Electrical and Computer Engineering (IJECE), Vol. 11, No. 1, February 2021, pp. 716–727.
doi: `10.11591/ijece.v11i1.pp716-727`.

[4] Luis Paquete, Carlos M. Fonseca, Manuel Lopez-Ibanez, Francisco Ruiz.
Subset selection on Pareto fronts: Algorithms and applications.
arXiv preprint arXiv:2002.11830, 2020.
Disponible en ligne : `https://arxiv.org/abs/2002.11830`

[5] Anne Auger, Johannes Bader, Dimo Brockhoff, Eckart Zitzler.
Hypervolume-based multiobjective optimization: Theoretical foundations and practical implications.
In *Theoretical Foundations of Evolutionary Computation*, GECCO 2009.
Version auteur disponible sur HAL : `https://hal.science/hal-00431274`

[6] Alexander Brownlee, Alain L. Dias, Richard Wood, Alan Chappell, John R. H. Rossiter, Jonathan M. Cooper.
Multi-objective optimization of cellular fenestration by an evolutionary algorithm.
Engineering Applications of Artificial Intelligence, Volume 43, 2015, Pages 23–32.
doi: `10.1016/j.engappai.2015.03.003`.

[7] Erhan Erkut and Susan Neuman.
Comparison of Four Models for Dispersing Facilities.
Location Science, Volume 1, Issue 3, Pages 199–214, 1991.

[8] Olivier Spanjaard. Cours 1 RHAD : Optimisation combinatoire multi-objectifs 1. Master IAD – RHAD, LIP6 – UPMC, 2012. `https://webia.lip6.fr/~spanjaard/cours/rhad12c1.pdf`.

[9] J. Huang, Z. Chen, and N. Dupin. Comparing local search initialization for k-means and k-medoids clustering in a planar Pareto Front, a computational study. In *Proceedings of the International Conference on Optimization and Learning*, pages 14–28. Springer, 2021.

[10] S. Sayın. Measuring the quality of discrete representations of efficient sets in multiple objective mathematical programming. *Mathematical Programming*, 87:543–560, 2000.