Elicitation incrémentale et recherche locale pour le problème de sélection multi-objectifs

---

MADMC 2019/2020

Sarah Lachiheb

21/01/2020

Table des matières

1 Introduction

Dans ce projet, nous allons explorer la notion d'apprentissage actif dans le domaine du multicritère sur le problème de sélection multi-objectifs. La procédure appliquée durant ce travail est l'apprentissage par élicitation incrémentale basée sur le regret. A partir de cette procédure, on souhaite restreindre l'espace des poids du décideur et ainsi réduire l'espace des solutions potentielles du décideur jusqu'à cerner suffisamment son vecteur poids pour résoudre le système de recommandation du problème de sélection multi-objectifs.

Le problème étudié et sur lequel nous allons tester deux algorithmes différents dans le but de faire une recommandation pertinente (à un décideur fictif) est le suivant :
Données :

- n objets
- k le nombre d'objets dans le sac, soit $k = \frac{n}{2}$ car les documents fournis nous permettent de comparer nos résultats.
- un vecteur permettant d'avoir la performance de l'objet sur chacun des critères, pour chacun des objets.

Solution : Les k objets que l'on choisit de prendre dans le sac.

But : Déterminer une solution réalisable maximisant une fonction d'agrégation paramétrée. Les paramètres représentent les préférences du décideur sur chacun des critères.

Dans un premier temps, nous utiliserons une méthode qui se constitue de deux phases. Une première où nous allons déterminer heuristiquement, un front de Pareto via une recherche locale. Puis nous appliquerons une procédure d'élicitation incrémentale dans le but de faire bonne recommandation au décideur.

Dans une seconde partie, nous allons étudier une méthode qui intègre l'élicitation à chaque étape de la recherche locale.

Les résultats sont données dans un premier temps avec un décideur modéliser par une somme pondérée. Puis un OWA à poids décroissant.
Comme il a été demandé dans l'énoncer, chaque résultat est issue d'une moyenne sur 20 itérations.

2 PLS puis élicitation incrémentale

2.1 Principe général

Algorithm 1 `Pareto Local Search Puis Elicitation incrémentale`

Paramètres $\downarrow$: P : population initiale aléatoire ; $\mathcal{N}(x)$: une fonction de voisinage ; Ψ_w : un agrégateur avec des paramètres inconnus ; Θ : un ensemble de préférences connues ; M : taille de la sélection pour la procédure d'élicitation incrémentale.
Paramètres $\uparrow$: x^* une solution à recommander

--| Détermination aléatoire d'une solution de départ :
$P \leftarrow$ generateSolution()
--| Initialisation de l'ensemble X :
$X \leftarrow \emptyset$
$P_{next} \leftarrow \emptyset$
while $P \neq \emptyset$ **do**
  --| On génère tous les voisins de chaque solution de P :
  for all $p \in P$ **do**
    for all $p' \in$ pareto($\mathcal{N}(p)$) **do**
      if p' n'est pas Pareto dominé **then**
        if Update(X, p') **then**
          Update(P_{next}, p')
        end if
      end if
    end for
  end for
  Mettre les nouvelles solutions Pareto trouvés :
  $P \leftarrow P_{next}$
  On vide l'ensemble P_{next} :
  $P_{next} \leftarrow \emptyset$
end while
if size(X) $> M$ **then**
  $X \leftarrow$ chooseRandom(X, M)
end if
while $MMR(X, \Omega_\Theta) > 0$ **do**
  --| Poser une question au décideur entre les deux alternatives qui minimisent de minimax regret :
  $(x, x') \leftarrow$ `Question`(X)
  --| On ajoute cette préférences à l'ensemble de celles connues :
  $\Theta \leftarrow \Theta \cup \{(x, x')\}$
  --| Cela réduit les paramètres possibles :
  $\Omega_\Theta \leftarrow \{\omega : \forall (x, x') \in \Theta, \Psi_w(x) \geq \Psi_w(x')\}$
end while
$x^* \leftarrow$ `Selection`($\arg\min_{x \in X} MR(x, X, \Omega_\Theta)$)
return x^*

L'algorithme 1 développé dans cette section possède une partie *PLS*, issue du cours numéro 4 de MADMC et une partie *Procédure d'élicitation incrémentale basée sur le regret* issue des deux articles fournis en annexes.
Dans une première partie, l'algorithme détermine une solution aléatoire et la place dans l'ensemble P. Ensuite, le voisinage 1-1 de cette solution est déterminé. Cela nous retourne par exemple pour $1-0-0-1$ le voisinage suivant :

$$0-1-0-1 \qquad 0-0-1-1 \qquad 1-1-0-0 \qquad 1-0-1-0$$

Une fois ce voisinage déterminé, on effectue un filtre de Pareto afin de ne garder que celles qui ne sont pas Pareto dominés. Enfin pour chaque solution de ce voisinage, si elle n'est pas dominée par un des éléments de P et seulement si elle est bien intégrée à l'ensemble X alors on l'ajoute à l'ensemble P_{next}. Cela nous permet de ne pas développer une solution qui aurait déjà été développée. En effet, le voisinage ne possèdent pas que des solutions qui "vont vers le front de Pareto", ainsi cette condition nous permet de ne pas effectuer de voisinage inutilement.
Une fois chaque élément de P développé, on ajoute les éléments de P_{next} dans P et on vide P_{next}. Ainsi, si P_{next} est vide alors P est alors vide et on stop la procédure. En effet, il n'y a plus de nouvelles solutions à développer.

Une illustration de cette méthodes est donnée en Figure 1 pour 20 objets et 2 critères. On peut voir l'étendue des voisinages à gauche ainsi que le déplacement du front de Pareto à droite. Cela nous permet de pouvoir voir à quelle itération chaque point du front de Pareto a été déterminé. On remarque, pour cette instance, que le front de Pareto est constitué dès l'itération 4, excepté une solution que l'on déterminera deux itérations après. Aucun nouveau point Pareto optimal n'a été trouvé à la sixième itération, et la procédure s'est alors arrêtée expliquant alors qu'il n'y ait aucun point de couleur verte (couleur des solutions déterminées à la dernière itération) dans le front de Pareto final.

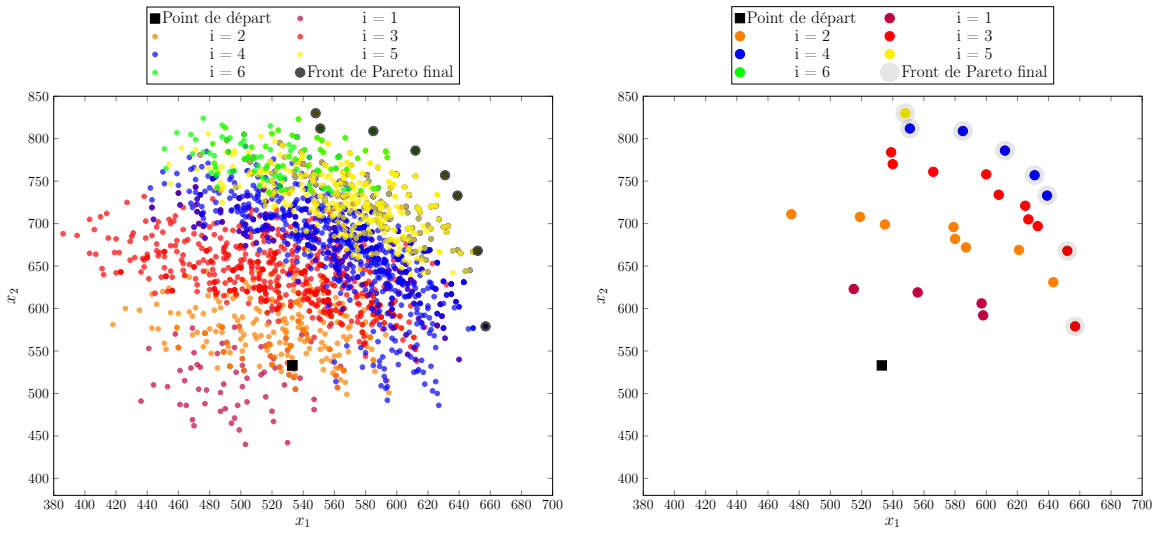

FIGURE 1 – Illustration de la PLS sur 20 objets et 2 critères, avec 6 itérations.

Une fois cette étape effectuée, il faut ensuite appliquer la procédure d'élicitation incrémentale. Elle est basée sur le minimax regret, en maximisation ici, et se déroule sur trois étapes dont le but est de poser une question au décideur entre deux solutions qui sont pires adversaires :

- $\text{PMR}(x, x', \Omega_\Theta) = \max_{\omega \in \Omega_\Theta} (f_\omega(x') - f_\omega(x)) = \max_{\omega \in \Omega_\Theta} \sum_{j \in \mathcal{P}} (\omega_j x'_i - \omega_j x_j)$
- $\text{MR}(x, X, \Omega_\Theta) = \max_{x' \in X} \{PMR(x, x')\}$
- $\text{MMR}(X, \Omega_\Theta) = \min_{x \in X} \{MR(x)\}$

Dans le but de calculer le PMR, on utilise un programme linéaire :

$$
\begin{array}{lll}
\max\limits_{\omega} & \sum\limits_{j \in \mathcal{P}} (\omega_j x'_j - \omega_j x_j) & \\
s.c & \sum\limits_{j \in \mathcal{P}} \omega_j = 1 & \\
 & \sum\limits_{j \in \mathcal{Q}} \omega_j a_j \geqslant \sum\limits_{j \in \mathcal{P}} \omega_j b_j & \forall a, b \in \Omega_\Theta \text{ avec } a \text{ préférée} \\
 & \omega_j \geqslant 0 & \forall j \in \mathcal{P}
\end{array}
$$

Auquel il faut ajouter une contrainte si on est dans le cadre d'un OWA à poids décroissant, dans le but de d'obliger à ce que les poids possibles respecte cette règle.

$$w_j - w_{j+1} \geqslant 0 \qquad \forall j \in \{1, \ldots, p-1\}$$

Ainsi, tant que le minimax regret des solutions de notre ensemble n'est pas égale à 0, alors on continue de poser des questions dans le but de réduire l'ensemble des jeux de poids en accord avec ses préférences. Cela représente aussi une réduction de l'espace des solutions que l'on pourrait recommander au décideur.

Par exemple, dans l'exemple utilisé, la taille du front de Pareto est de 8. Et la procédure doit poser 3 questions dans le but d'avoir la garantie de la solution recommandée est celle qui est optimale. On peut observer les questions qui sont posées en Figure 2.

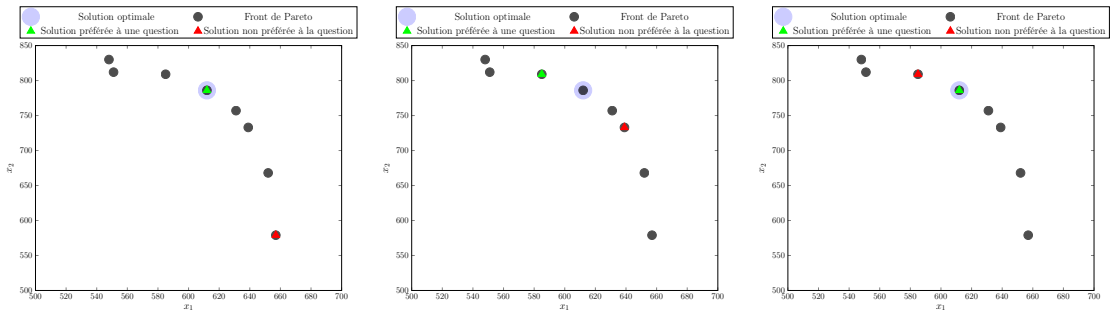

FIGURE 2 – Illustration de la procédure d'élicitation incrémentale sur 20 objets et 2 critères, avec 3 questions.

En Figure 3 on peut observer la valeur du minimax regret en fonction des questions posées.

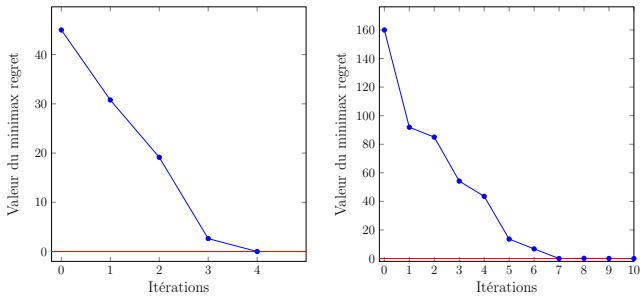

FIGURE 3 – Illustration des valeurs de minimax regret en fonction de l'itération pour 20 objets 2 critères à gauche et 30 objets et 3 critères à droite.

On observe que la valeur du minimax regret ne fait que diminuer avec le nombre de questions. En effet, chaque question réduit le l'espace des paramètres possibles donc cela réduit le plus petit des max regret de prendre une solution à la place d'une autre car elles sont toutes plus proches de la solution optimale.

Cela est plus d'autant plus visible en Figure 3 à droite, où l'on à utilisé le front de Pareto (issue de la PLS correspondante) de 30 objets et 3 critères, contenant 315 alternatives. Il est nécessaire de poser 9 questions pour obtenir MMR = 0, cependant on remarque que les valeurs du MMR à l'itération 7, 8 et 9 sont très proches de 0 soit : 2.63, 0.14, et 0.03. On peut observer que la certitude de la procédure nécessite beaucoup de question comparé à ce qu'il pourrait être fait si la condition de 0 était détendue, comme dans l'article donné en Annexe.

2.2 Résultats avec la somme pondérée

On donne les résultats en Table 1 de la PLS qui permet de définir une approximation du front de Pareto. On remarque que le voisinage 1-1 est très efficace car il permet, en partant de n'importe quelle solution, d'aller, par succession échanges 1-1, à n'importe quelle autre solution. Cela nous permet d'obtenir tous les points du front de Pareto en comparaison à ceux fourni dans ce projet. Les tests sont uniquement fait pour K, le nombre d'objets dans le sac, égale à la moitié de la taille de l'instance. Cependant les temps de cette méthode sont longs, et si les dimensions du problème devenait plus grande, alors il deviendrait impossible de générer le front de Pareto de cette manière. On remarque aussi que les itérations variables selon le point de départ. Ainsi, on pourrait avoir une amélioration de cette méthode en partant d'un bon point de départ pour la PLS.

En Figure 4 on montre l'incidence de l'augmentation du nombre de critères et de la taille de l'instance. Plus le nombre de critères est élevé et plus le nombre de question et le temps d'exécution seront élevés. Parallèlement à la taille de l'instance.

Taille	n	Temps(s)	Itérations	Taux d'identiques
10	2	0.0003	3.8	100
10	3	0.0006	4.4	100
10	4	0.0010	4.9	100
10	5	0.0047	5.3	100
10	6	0.0044	5.3	100
20	2	0.0061	7.3	100
20	3	0.0380	7.8	100
20	4	0.3721	8.3	100
20	5	6.7656	9.1	100
20	6	22.4597	9.3	100
30	2	0.0459	10.0	100
30	3	0.6250	10.9	100
30	4	36.7146	11.7	100
30	5	2064.9400	11.5	100
40	2	0.2128	12.2	100
40	3	4.5793	13.6	100
40	4	364.2876	14.2	100
50	2	1.0233	15.2	100
50	3	22.1369	16.1	100
60	2	3.7206	18.1	100
60	3	173.9490	20.4	100
80	2	19.2443	22.3	100
100	2	90.4203	28.0	100
200	2	6318.2701	55.1	100

Taille	p	Temps(s)	Questions	Écart à l'optimal
10	2	0.019	1	0
10	3	0.171	2.5	0
10	4	1.015	8.1	0
10	5	2.791	10.1	0
10	6	3.435	11.6	0
20	2	0.179	2.8	0
20	3	4.276	7.0	0
20	4	156.309	16.1	0
20	5	-	-	-
20	6	-	-	-
30	2	0.350	2.7	0
30	3	22.314	8.6	0
30	4	-	-	-
30	5	-	-	-
40	2	0.830	2.7	0
40	3	59.222	10.4	0
40	4	-	-	
50	2	1.381	4.1	0
50	3	417.767	9.6	0
60	2	5.397	5.3	0
60	3	-	-	-
80	2	15.847	6.1	0
100	2	36.463	6.2	0
200	2	153.033	6.5	0

TABLE 1 – Performances, de PLS à gauche et de la procédure d'élicitation incrémentale basée sur le regret à droite.

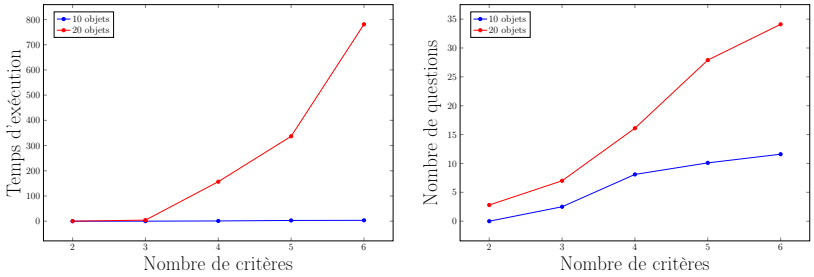

FIGURE 4 – Temps d'exécution et nombre de questions en fonction du nombre de critères pour 10 objets et 20 objets.

Du fait que la méthode de voisinage soit très performante, l'ensemble des points Pareto optimaux peut se trouver être très grand. Ainsi, il devient trop long pour la méthode d'élicitation incrémentale basée sur le regret. Par exemple, pour 20 objets et 6 critères la taille est de 4315 ou de 32658 pour 30 objets et 5 critères. Pour palier à ce problème, il est proposé (suite à un échange de mails avec Monsieur Thibaut Lust) de choisir, aléatoirement, un nombre M de solutions Pareto optimales. Ainsi, dans le tableau Table 2, pour les fronts de Pareto excédant une taille de M, on exécute la méthode d'élicitation sur M alternatives. Ainsi, la distance à l'optimale sera impactée car il est possible que la solution optimale ne soit pas dans la sélection, mais l'algorithme pourra ainsi recommander une solution au décideur.

Logiquement, on constate que plus M est grand plus la procédure est longue, plus le nombre de question est élevé et plus l'écart à l'optimum est faible. Nous disposons d'aucune garantie sur les résultats cependant, en pratique ils sont très bons, moins de 1% d'erreur avec $M = 1000$ et moins de 2% avec $M = 100$.

Taille	n	Temps(s)	Questions	Écart à l'optimal	Taux du Front de Pareto(%)
20	5	330.062	27.9	0.27	36.14
20	6	758.470	34.1	0.38	23.18
30	4	676.463	14.9	0.17	20.59
30	5	463.256	26.4	0.65	3.06
40	4	536.551	15.9	0.34	7.71
60	3	582.222	9.80	0.06	31.37

TABLE 2 – Performances de la procédure d'élicitation incrémentale basée sur le regret avec $M = 1000$.

Taille	n	Temps(s)	Questions	Écart à l'optimal	Taux du Front de Pareto(%)
20	5	6.413	14.7	1.25	3.61
20	6	14.306	19.1	1.88	2.31
30	4	12.745	12.2	1.12	2.06
30	5	8.049	19.80	1.94	0.31
40	4	7.021	15.3	0.75	0.77
60	3	6.683	7.4	0.27	3.14

TABLE 3 – Performances de la procédure d'élicitation incrémentale basée sur le regret avec $M = 100$.

Il peut sembler surprenant, malgré le fait que l'on ne prennent qu'un pourcentage inférieur à 5% du front de Pareto, avec $M = 100$, on ait des résultats si proches de l'optimal (en Table 3), soit moins de 2%. Cependant, statistiquement, cela semble normale. Par exemple pour 60 objets et 3 critères, sur 20 exécutions et selon les jeux de poids, il y a une moyenne de 9.97% de solutions inférieures à 1% d'erreur. Cet ensemble est représenté en Figure 5 pour le jeu de poids $w = (0.2, 0.3, 0.5)$. Étant donné que l'on en choisit 100, il est donc normal d'avoir de manière quasi-certaine, au moins une solution de distance inférieure à 1% de l'optimal.

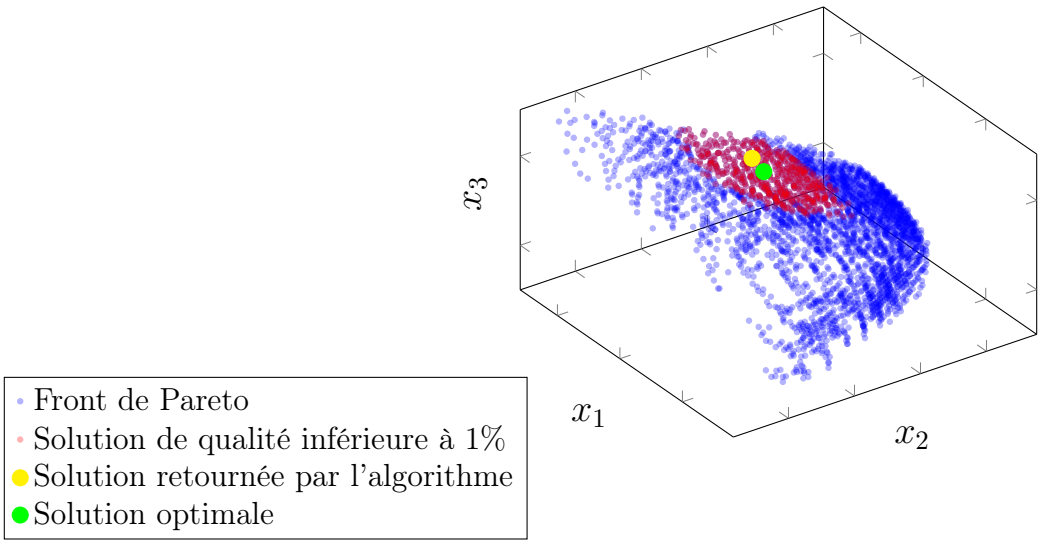

FIGURE 5 – Illustration des solutions inférieures à 1% d'erreur pour 60 objets et 3 critères avce le poids caché (0.2, 0.3, 0.5).

2.3 Résultats avec un OWA à poids décroissants

On présente dans cette sections les mêmes expérimentations que dans la partie précédente, mais avec un OWA à poids décroissants. C'est à dire que le jeux de poids du décideur possède des valeurs décroissantes. Aussi, il faut trier les valeurs des solutions dans l'ordre décroissant. Le calcul du PMR est aussi impacté car il faut ajouter la contrainte de poids décroissants. Cependant, ici nous utilisons les fronts de Pareto déterminés avec l'algorithme PLS.

Taille	p	Temps(s)	Questions	Écart à l'optimal
10	2	0.021	0	0
10	3	0.090	2.0	0
10	4	0.385	2.0	0
10	5	1.653	4.8	0
10	6	1.006	1.3	0
20	2	0.069	1.2	0
20	3	2.955	3.8	0
20	4	146.531	6.6	0
20	5	-	-	-
20	6	-	-	-
30	2	0.317	2.6	0
30	3	7.139	3.1	0
30	4	-	-	-
30	5	-	-	-
40	2	0.820	3.0	0
40	3	24.753	5.8	0
40	4	-	-	
50	2	1.582	2.9	0
50	3	276.320	5.9	0
60	2	7.627	3.2	0
60	3	-	-	-
80	2	13.748	2.3	0
100	2	23.730	2.8	0
200	2	273.214	3.7	0

Taille	n	M	Temps(s)	Questions	Écart à l'optimal
20	5	1000	105.467	5.9	0.29
20	6	1000	77.185	3.5	0.46
30	4	1000	205.975	6.4	0.48
30	5	1000	66.657	4.0	0.93
40	4	1000	244.880	7.6	0.42
60	3	1000	50.519	2.7	0.19
20	5	100	2.995	4.2	1.75
20	6	100	1.761	2.3	2.24
30	4	100	2.990	3.8	1.51
30	5	100	2.12	3.1	2.35
40	4	100	3.106	4.3	1.04
60	3	100	1.395	1.6	0.87

TABLE 4 – Performances, de PLS à gauche et de la procédure d'élicitation incrémentale basée sur le regret à droite.

On remarque que le nombre de questions est nettement inférieur, par exemple seulement 5.8 contre 10.4 questions pour 40 objets et 3 critères avec la somme pondérée. Cela s'explique par une grande réduction de l'ensemble des jeux de poids possibles. En effet, nous ajoutons des contraintes au calcul du PMR, ce qui à pour conséquence de réduire l'ensemble des jeux de poids possibles. Cela agit de la même manière que si nous avions déjà collecté des informations auprès du décideur dans le cas de la somme pondérée et on saurait que $w_{i+1} \leq w_i \forall i \in \mathcal{P}$. Aussi, nous avons vu en cours de MADMC que le nombre de solutions optimales pour un OWA à poids décroissants est inférieur aux nombre de solutions optimales avec une somme pondérée. Logiquement, on peut penser que cela aide aussi à réduire le nombre de questions à poser car il y a moins de solutions optimales avec ces critères supplémentaires.

En supplément, on peut raisonnablement penser que comme le nombre de solutions optimales avec une Intégrale de Choquet est supérieur à celui avec une somme pondérée, alors le nombre de questions avec une Intégrale de Choquet serait encore supérieur à celui avec une somme pondérée.

Cette partie nous permet de comprendre l'utilité de bien choisir sa fonction d'agrégation. En effet, si nous choisissons une Somme pondérée ou un Choquet pour modéliser

un décideur et que cela nous retourne une solution OWA-optimale à poids décroissant alors de nombreuses questions posées au décideur ne sont pas nécessaire.
Toujours dans le but de limiter le nombre de questions à poser, il est important de déterminer la meilleure modélisation.

Dans une seconde partie nous allons étudier une méthode différente dans laquelle la recherche locale et élicitation incrémentale basée sur le regret sont couplées.

3 PLS combiné à la procédure d'élicitation incrémentale

3.1 Principe général

Algorithm 2 `Elicitation Et Recherche Locale`

Paramètres ↓ : p : solution initiale aléatoire ; $\mathcal{N}(p)$: une fonction de voisinage ; Ψ_w : un agrégateur avec des paramètres inconnus ; Θ : un ensemble de préférences connues ; K : taille de la sélection pour la procédure d'élicitation incrémentale.
Paramètres ↑ : x_* une solution à recommander

--| Détermination aléatoire d'une solution de départ, solution courante :
$p \leftarrow$ generateSolution()
--| Initialisation de la développée à l'itération précédente :
$x_* \leftarrow \emptyset$
while $p \neq x_*$ **do**
  --| On génère les voisins de p et on ne conserve que les Pareto optimales :
  $P \leftarrow pareto(\mathcal{N}(p) + p)$
  while $MMR(X, \Omega_\Theta) > 0$ **do**
    --| Poser une question au décideur entre les deux alternatives qui minimisent de minimax regret :
    $(x, x') \leftarrow$ `Question`(P)
    --| On ajoute cette préférences à l'ensemble de celles connues :
    $\Theta \leftarrow \Theta \cup \{(x, x')\}$
    --| Cela réduit les paramètres possibles :
    $\Omega_\Theta \leftarrow \{\omega : \forall (x, x') \in \Theta, \Psi_w(x) \geq \Psi_w(x')\}$
  end while
  $x_* \leftarrow p$
  $p \leftarrow$ `Selection`$(\arg\min_{x \in P} MR(x, P, \Omega_\Theta))$
end while
return x_*

Dans cette section et comme décrit dans l'algorithme 2, la procédure ne se fait plus en deux étapes mais une seule. Maintenant, après avoir choisit une solution de manière aléatoire, on détermine sont voisinages, et dans ce voisinage, à l'aide de la procédure d'élicitation incrémentale basée sur le regret, on détermine quelle est la préférée du décideur dans cet ensemble. Ensuite cette solution devient la solution courante et on

la développe à son tour. On effectue cette boucle jusqu'à ce que la même solution soit encore sélectionnée en tant que préférée du décideur.

Pour plus de visualisation, on peut voir en Figure 6 ma représentation de cette méthode pour 20 objets et 2 critères. Il y a 6 itérations pour seulement 2 questions.

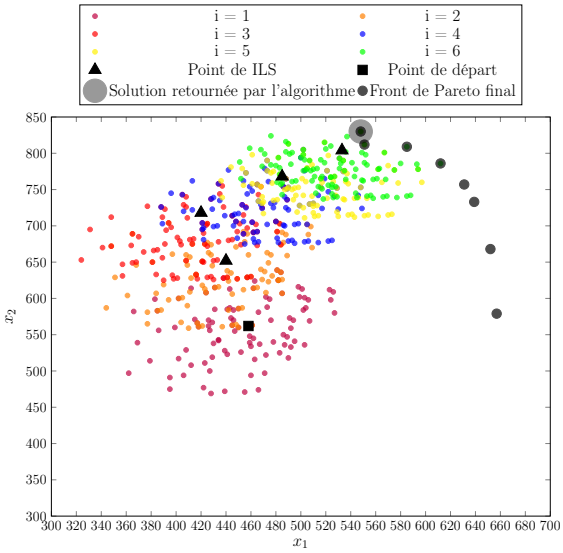

FIGURE 6 – Illustration de ILS sur 20 objets et 2 critères, avec 6 itérations.

On voit que contrairement à la méthode précédente, nous avons pas besoin ici d'explorer le font de Pareto dans toutes les direction. On fait donc une économie de parcours et donc on se dirige, dès la première itération, dans une direction qui est intéressante pour le décideur. Intuitivement, on peut penser que cette méthode peut avoir de meilleures résultats que la précédente.

3.2 Résultats avec la somme pondérée

En effet, on observe, en Table 6, que les résultats avec ILS sont bien meilleures que la méthode en deux phases. Aussi, comme nous explorons qu'un seul voisinage à la fois, les dimension sont beaucoup plus raisonnable et les temps sont réduit. Aussi, grâce à l'efficacité de cette méthode qui se dirige directement vers la solution préférée du décideur, elle pose moins de question.
Elle nous permet aussi de pouvoir effectuer des tests là où précédemment nous avions dû sélectionner qu'une partie du front de Pareto à cause de sa taille.

Pareillement à la Figure 4, la Figure 7 montre l'incidence de l'augmentation du nombre de critères et de la taille de l'instance. On peut y faire les mêmes conclusion bien que les valeurs de deviennent pas aussi élevées qu'avec l'algorithme précédent.

Taille	p	Temps(s)	Questions	Écart à l'optimal	Nombre d'itérations
10	2	0.019	0.4	0	3.8
10	3	0.058	2.2	0	3.6
10	4	0.193	5.8	0	3.7
10	5	0.377	8.9	0	3.5
10	6	0.555	9.8	0	3.4
20	2	0.068	2.4	0.02	6.0
20	3	0.517	8.15	0	5.9
20	4	1.901	13.4	0.02	6.1
20	5	4.836	18.8	0.002	5.8
20	6	6.802	22.3	0	5.9
30	2	0.125	3.3	0	8.7
30	3	1.264	10.8	0	8.4
30	4	8.065	19.5	0	8.7
30	5	17.117	26.6	0	8.9
40	2	0.199	3.0	0	10.8
40	3	2.062	10.4	0	10.8
40	4	11.101	22.5	0	11.2
50	2	0.311	3.6	0	13.6
50	3	3.248	11.2	0	13.5
60	2	0.562	4.1	0	15.7
60	3	5.311	13.5	0	15.9
80	2	1.001	4.2	0	21.8
100	2	1.459	4.9	0	24.9
200	2	15.394	5.7	0	51.3

TABLE 5 – Performances de ILS avec une somme pondérée.

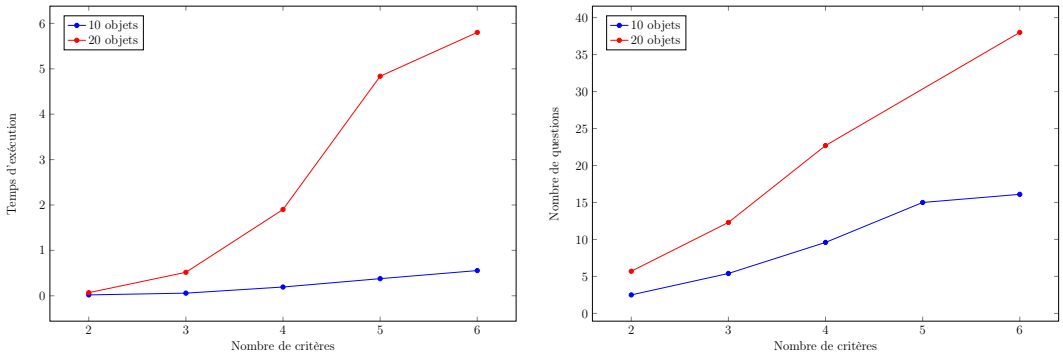

FIGURE 7 – Temps d'exécution et nombre de questions en fonction du nombre de critères pour 10 objets et 20 objets.

3.3 Résultats avec un OWA à poids décroissants

On peut faire les mêmes conclusions que dans la première partie. En effet, on observe que avec la modélisation par un OWA nous posons toujours considérablement moins de questions qu'avec une somme pondérée. Par exemple, pour 60 objets et 3 critères, on pose en moyenne 13.4 questions en $5.3s$, alors qu'avec un OWA on pose seulement 4.5 questions en $2.120s$ pour une qualité similaire. Cependant, la différence avec la solution optimale du décideur est plus grande qu'avec une somme pondérée, même si elle reste très proche et aussi sans garantie.

Taille	p	Temps(s)	Questions	Écart à l'optimal	Nombre d'itérations
10	2	0.005	0.4	0	3.4
10	3	0.032	2.8	0	3.5
10	4	0.058	3.2	0	3.7
10	5	0.109	5.7	0	3.4
10	6	0.163	4.9	0	3.7
20	2	0.046	4.1	0	5.9
20	3	0.380	8.9	0	6.3
20	4	0.823	7.5	0	5.9
20	5	0.842	12.0	0	6.2
20	6	1.397	6.5	0	6.4
30	2	0.088	1.2	0	9.0
30	3	0.510	3.2	0.03	8.1
30	4	1.983	7.8	0.47	8.5
30	5	2.711	8.1	0.54	8.1
40	2	0.165	1.5	0	10.9
40	3	1.322	6.7	0	10.1
40	4	4.391	11.5	0.23	11.4
50	2	0.277	1.6	0	13.1
50	3	2.971	7.6	0.01	14.2
60	2	0.386	2.4	0.05	16.8
60	3	2.120	4.5	0.03	16.5
80	2	0.835	1.8	0.10	22.1
100	2	1.015	1.9	0.05	25.1
200	2	14.211	2.3	0.12	49.6

TABLE 6 – Performances de ILS avec un OWA.

4 Comparaisons entre les deux méthodes

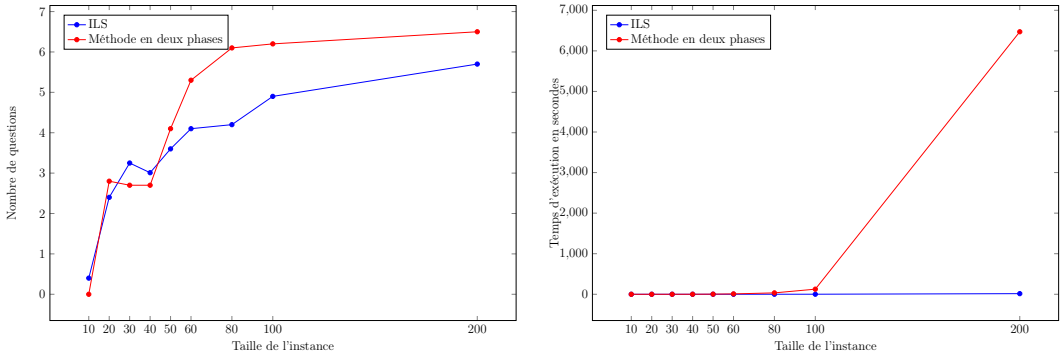

FIGURE 8 – Comparaison pour 2 critères des deux méthodes, du nombre de questions à droite et du temps d'exécution à gauche, en fonction de la taille de l'instance.

Pour un nombre de critères égale à 2, en Table 8, et pour une taille d'instance importante, supérieur à 40, on constate que ILS est la meilleure méthode à utiliser car elle pose moins de questions (cependant les valeurs restent très proches) en un temps qui est très inférieur.

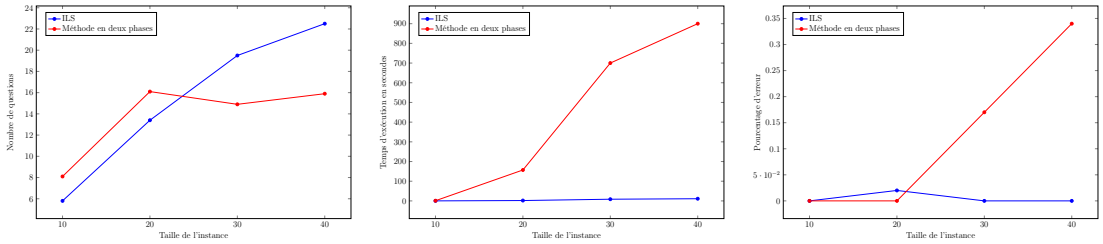

FIGURE 9 – Comparaison pour 4 critères des deux méthodes, du nombre de questions à droite et du temps d'exécution au centre, et du pourcentage d'erreur à gauche, en fonction de la taille de l'instance.

Pour un nombre de critères élevé, en Table 9, la comparaison est plus délicate car étant donné que l'exécution de la procédure d'élicitation n'était pas possible à cause de la taille des fronts de Pareto avec la méthode en deux phases, on a choisie de tronquer l'ensemble à partir de 30 objets pour 4 critères (ce qui explique la décroissance de la courbe du nombre de question de la procédure). Aussi, cela nous retourne des résultats plus éloignés de l'optimal (en pratique ils restent très proches) comparé à ILS. Cependant, les temps d'obtention des fronts de Pareto peuvent être un problème pour l'utilisation de la méthode en deux phases comparativement à la rapidité de ILS où nous avons pu faire toutes les expérimentations.

Il peut être intéressant de remarquer que malgré la différence de méthode, le nombre d'itérations de la recherche locale reste équivalent. Par exemple, avec PLS nous faisons en moyenne 22.3 itérations pour 80 objets et 2 critères et avec ILS 21.8. On peut en déduire qu'en dépit de la méthode, le nombre de voisinage à exploré pour atteindre une solution du front de Pareto reste environ le même.

5 Conclusion

Dans ce projet nous avons étudier deux méthodes différentes qui ont le même but, soit recommander une solution pertinente au décideur. Ces algorithmes sont basés sur une heuristique avec voisinage et un procédure d'élicitation incrémentale basée sur le regret nous permettant d'obtenir, de manière garantie, la meilleure solution d'un ensemble (pour notre décideur et selon son jeux de poids inconnu). Ce projet m'a permis de mieux comprendre ce que peut être un système de recommandation, ainsi que l'importance du choix de la fonction d'agrégation pour modéliser de décideur.
Dans le but d'étendre ce projet il aurait été intéressant de partir d'un point de départ non aléatoire, comme cela est fait dans l'article donné en Annexe. Aussi, une implémentation de l'intégrale de Choquet pour modéliser le décideur nous aurait permis de pouvoir comparer les résultats face à un OWA à poids décroissant et à une somme pondérée. Nous pourrions aussi utiliser une autre fonction de voisinage pour la recherche locale.