Rapport Projet MADMC

David Pinaud, Mickael Hamdad

Master 2 : A.N.D.R.O.I.D.E, Sorbonne Université

Janvier 2022

TABLE DES MATIERES

1. **Première procédure de résolution**

1.1. PLS

PLS (PLS.py) est initialisé avec une population initiale composée de solutions dont les objets sont choisis parmi ceux qui maximisent un rapport de performance (somme pondérée des critères sur les poids) tant que la somme des poids des objets choisis ne dépassent pas le poids total W autorisé.

Il utilise une fonction de voisinage qui fait des échanges 1-1 et remplit l'espace qui reste avec des objets qui peuvent rentrer dans le sac.

Une tentative d'utilisation d'un QUAD-Tree (quad_tree.py) a été codée mais n'a pas eu de résultats satisfaisants.

Deux variantes de PLS ont été codées, la différence entre la première et la deuxième version est que la deuxième gère de façon plus intelligente l'ensemble des approximations des optimums de pareto.

Les exécutions de PLS sont enregistrés dans ./logs. Toutes les combinaisons d'instances avec un nombre d'objets **n** de 1 à 70 avec un nombre de critères **p** de 1 à 5 (jusqu'à 4 pour les instances où le nombre d'objets est supérieur à 48 pour une question de temps) ont été calculées et loggés.

Ces logs contiennent :

- la fonction PLS utilisée
- l'approximation de l'ensemble des non dominés
- le nom de la fonction de voisinage utilisée
- l'instance du problème considéré
- la capacité max du sac à dos
- le temps d'exécution
- le nombre d'objets
- le nombre de critères.

Un fichier *instance_loader.py* permet de récupérer des instances étant donné un nombre d'objets et de critères donné.

1.2. Élicitation incrémentale

Trois agrégateurs ont été codés : la somme pondérée (fichier elicitation_ponderee.py), OWA (fichier elicitation_OWA.py) et l'intégrale de Choquet (fichier elicitation_choquet.py).

On peut lancer l'élicitation incrémentale des préférences pour un décideur choisi au hasard (le décideur est soit une liste de poids, soit une capacité). On fait l'hypothèse qu'on connaît un certain nombre de préférences du décideur qui vont permettre une restriction initiale de l'espace des paramètres possibles.

Le calcul des Pairwise-Max-Regret (PMR) est fait sous Gurobi grâce aux programmes linéaires fournis dans l'article 1. On arrête l'élicitation lorsque le Min-Max-Regret (MMR) est inférieur à 0 (on a aussi la possibilité de fournir un certain seuil d'arrêt)

Lancer une élicitation incrémentale génère un log dans les dossiers *./logs_SP, ./logs_OWA* et *./logs_choquet* pour la somme pondérée, OWA et l'intégrale de choquet respectivement. Les logs ont été calculés pour toutes les combinaisons de n de 1 à 25 et pour p de 1 à 4.

Ces logs contiennent :

- Le fichier log PLS à partir duquel on a récupéré l'instance et l'approximation de l'ensemble des non dominés
- Les vecteurs performances de l'approximation des non dominées
- La solution optimale estimée par l'élicitation *OPT_elici*
- Le nombre de questions posé
- La valeur de la fonction objectif de *OPT_elici*
- Les paramètres (réels) du décideur
- La solution optimale calculée à partir des poids du décideur sur l'approximation des non dominées *OPT_X*
- La valeur de la fonction objectif de *OPT_X*
- La durée de l'élicitation
- La différence en pourcentage (gap) entre la valeur de la fonction objectif de *OPT_X et OPT_elici*
- La solution optimale calculée à partir des poids du décideur sur l'instance réelle *OPT*
- La valeur de la fonction objectif de *OPT*
- La différence en pourcentage (gap) entre la valeur de la fonction objectif de *OPT et OPT_elici*

À propos des paramètres du décideur, pour OWA et la somme pondérée, c'est une liste de réel entre 0 et 1 et sommant à 1. Pour l'intégrale de Choquet, on utilise un objet Capacite. On peut générer une capacité convexe aléatoire.

2. Deuxième procédure de résolution

La fonction de voisinage est la même que celle utilisée dans PLS. On utilise les fonctions d'élicitation codées dans la partie 1.2.

Des logs sont générés dans le dossier *./logs_RL_elici.py*. Toutes les itérations sont enregistrées, décrites :

- La solution courante
- Sa valeur
- Son vecteur performance,
- Le nombre de questions posées,
- La durée de l'élicitation et celle de l'itération

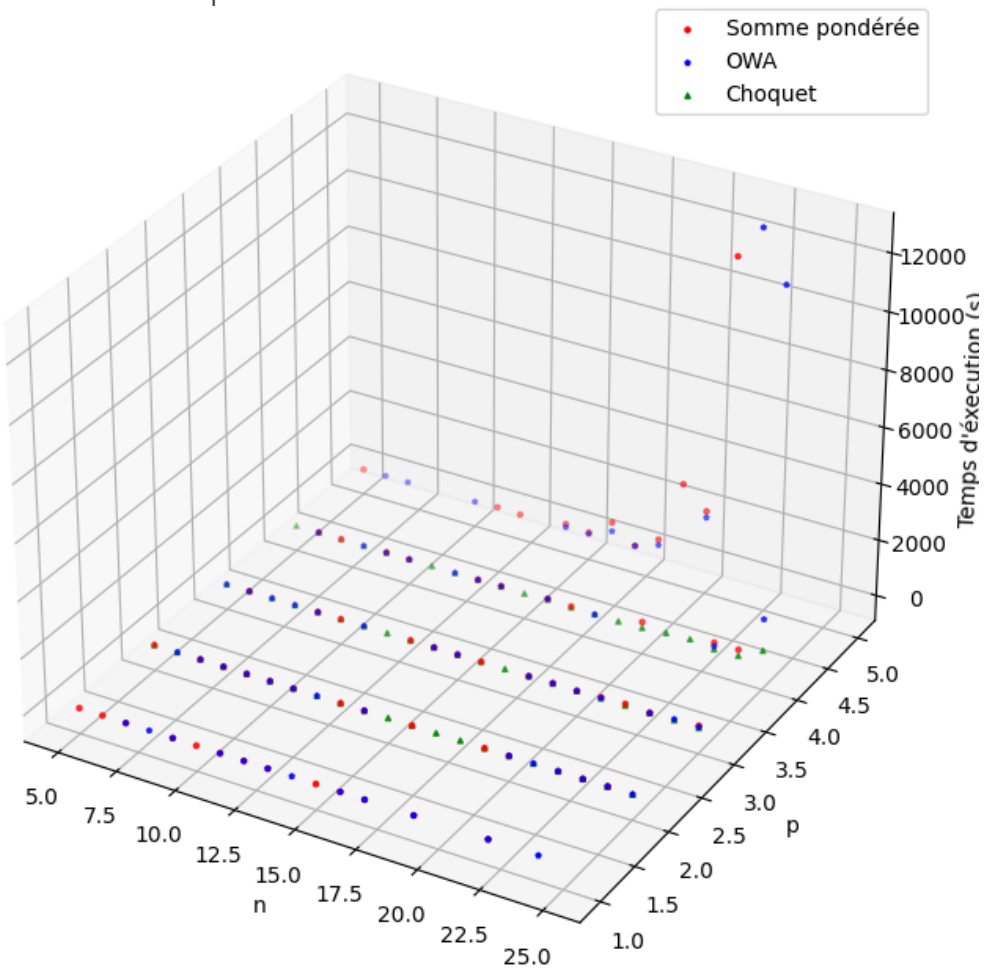
La solution initiale, la fonction voisinage, les paramètres du décideur ainsi que l'instance considérée sont enregistrés. Des logs ont été générés pour n de 2 à 25 et pour p de 2 à 4.

3. Graphes et comparaisons

À partir des logs, des graphes et statistiques ont été réalisés. D'autres logs dans les dossiers finissant par "pour_moyenne" ont été réalisés avec n=20 et p=4 et un minimum de 20 exécutions pour cette instance là. Ces logs vont nous permettre d'avoir des statistiques moyennés sur les différentes procédures implémentées.

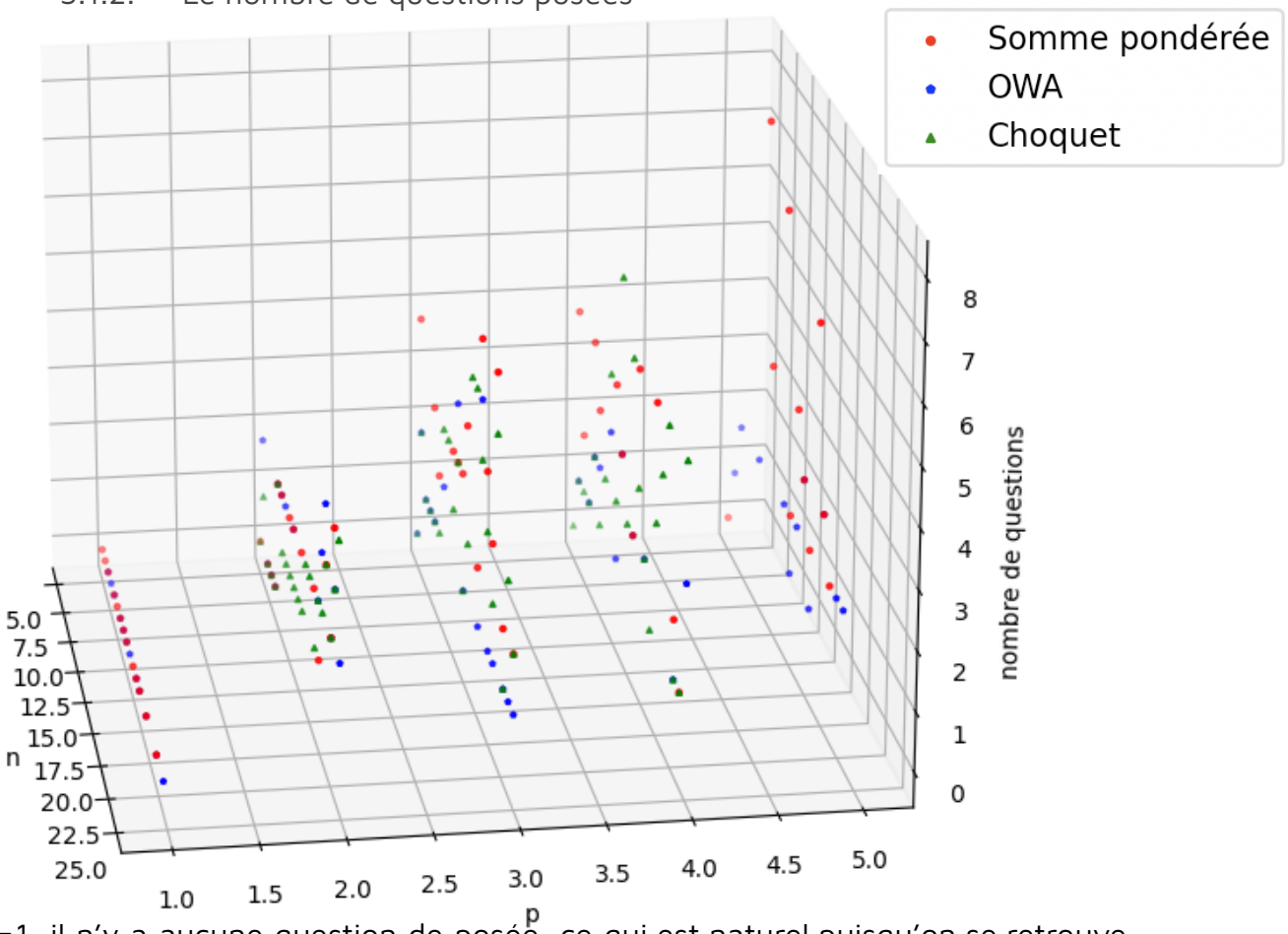
3.1. Première procédure de résolution

3.1.1. Temps d'exécutions



Nous pouvons voir que pour les trois agrégateurs différents, les temps d'exécutions sont minimes pour p<4 mais commencent à augmenter exponentiellement lorsque p≥5. Peut-être faudrait-il tester pour des valeurs de n plus grandes pour savoir quand est-ce que n aura un impact.

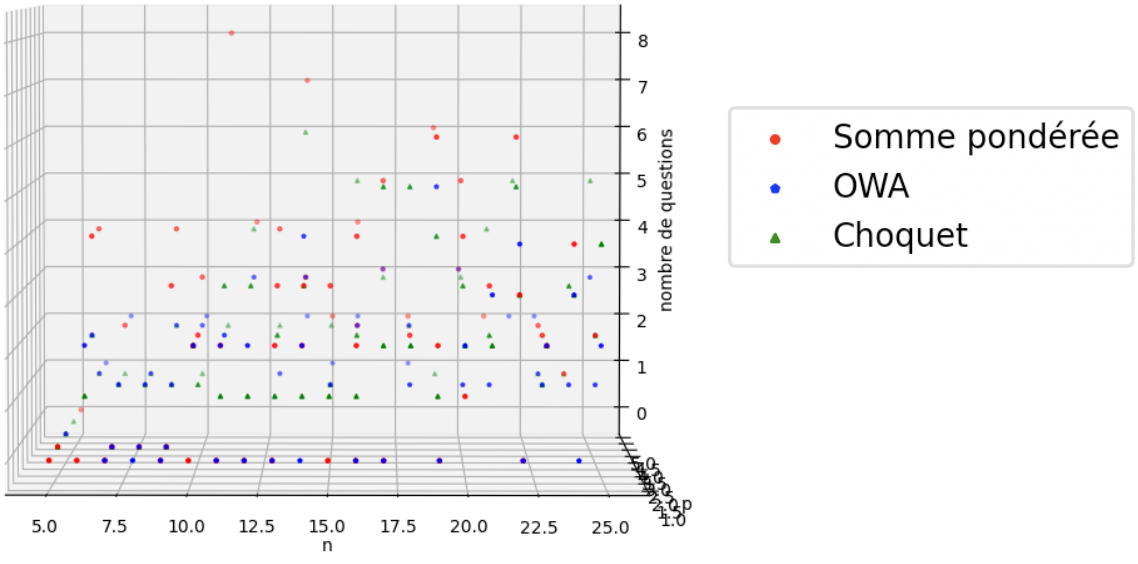
3.1.2. Le nombre de questions posées



Pour p=1, il n'y a aucune question de posée, ce qui est naturel puisqu'on se retrouve à faire de l'optimisation mono-objectif.

Pour p≥2, nous voyons une augmentation progressive du nombre de questions. Celui ci semble augmenter légèrement aussi avec la valeur de n:

(C'est le même graphe sous un autre angle)

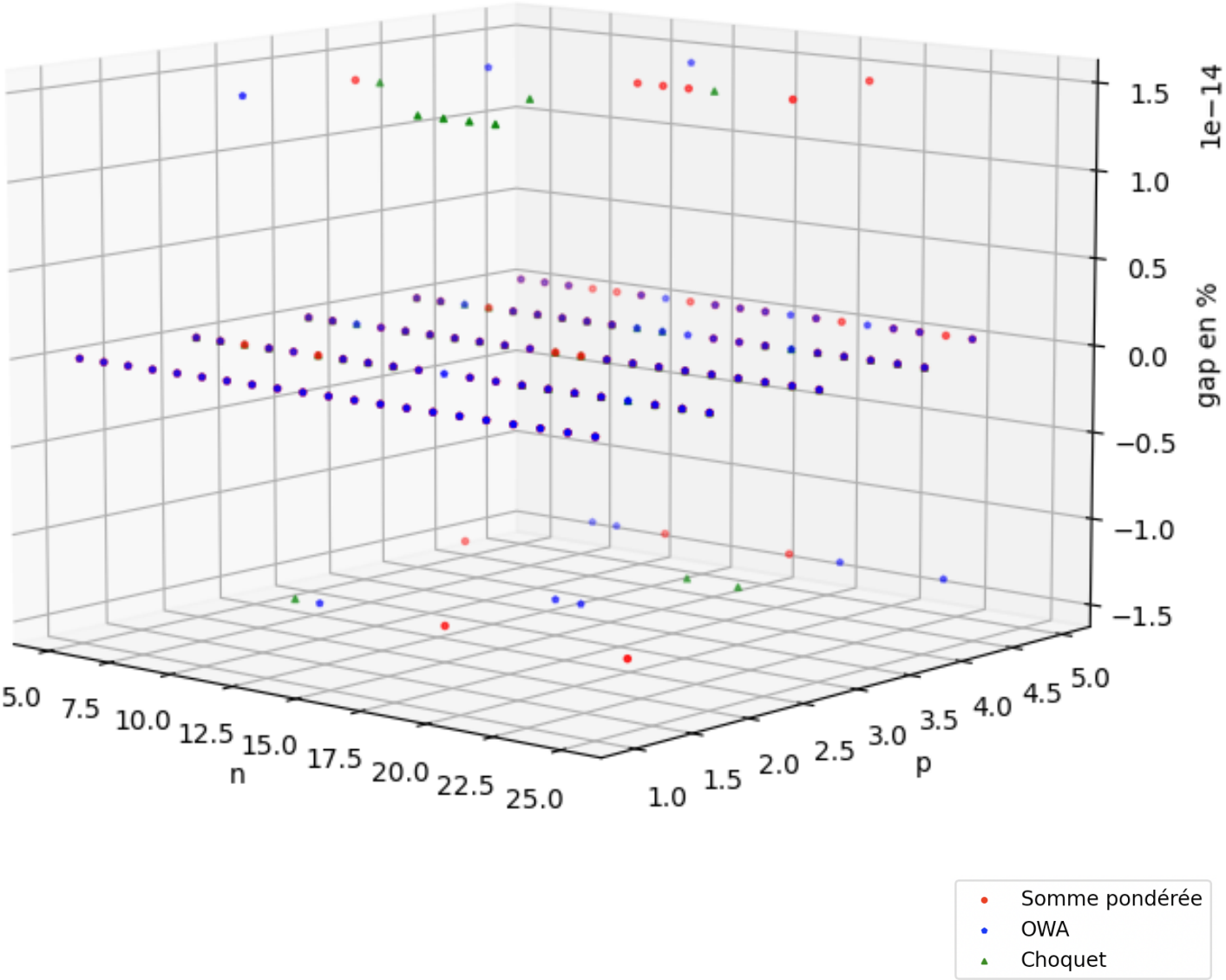


OWA pose moins de questions que les deux autres agrégateurs (nous retrouverons cette déclaration dans les statistiques) et la somme pondérée en pose le plus.

3.1.3. Gap de la valeur de la fonction objectif

Nous avons comparé la solution obtenue grâce à l'élicitation avec deux autres solutions : la solution optimale parmi les solutions dans l'ensemble des non-dominés rendu par PLS et la solution optimale de l'instance du problème calculé grâce à la résolution par programmation linéaire.

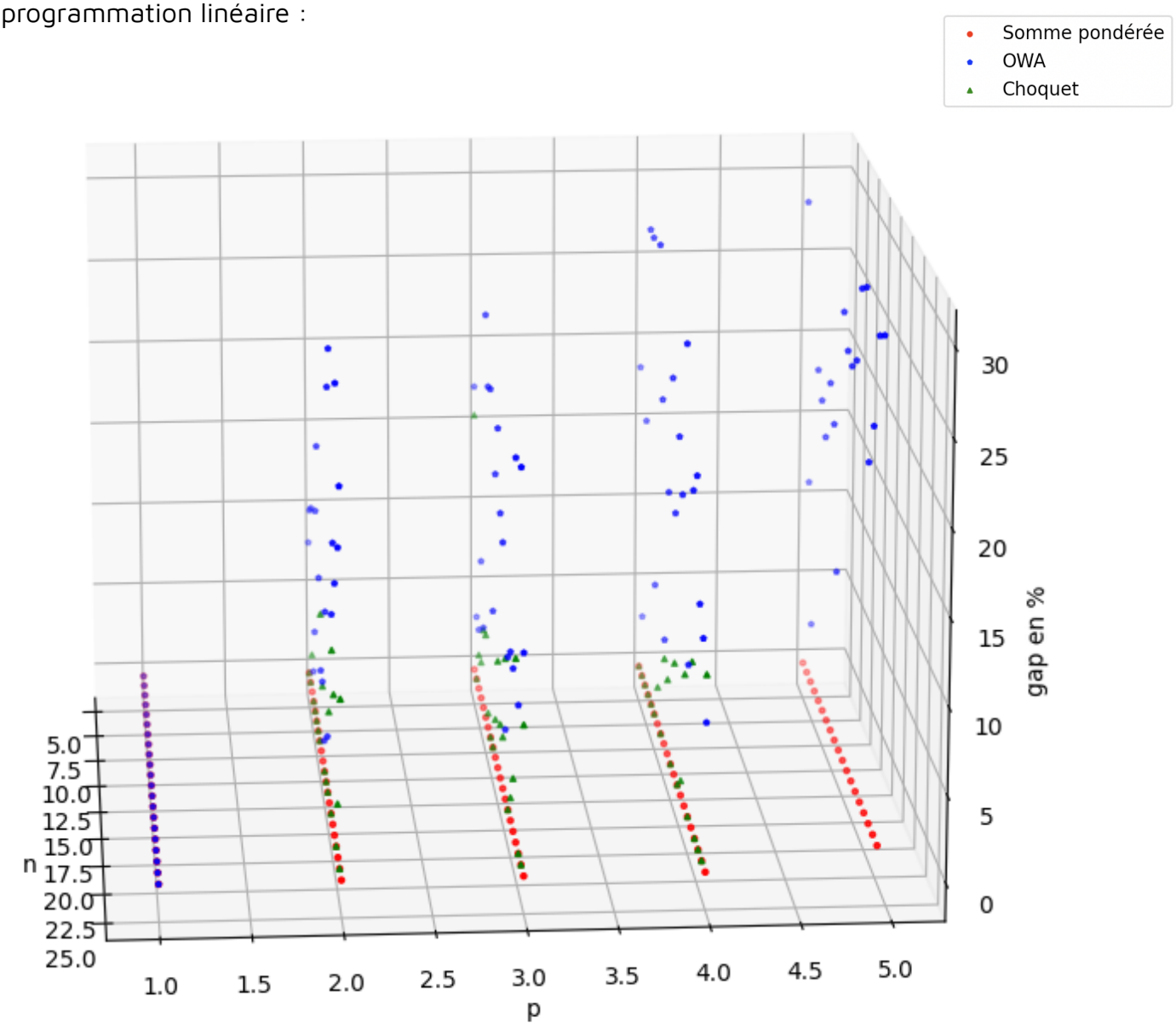
Face à la solution optimale calculé parmi les solutions dans l'ensemble des non-dominés :



Le gap est très petit, en réalité, il est nul. Les points non nuls sont très proches de 0 et l'erreur est dû à la représentation de réel dans les ordinateurs. On conclut donc

que l'élicitation permet bien de trouver la meilleure solution parmi celles proposées par PLS.

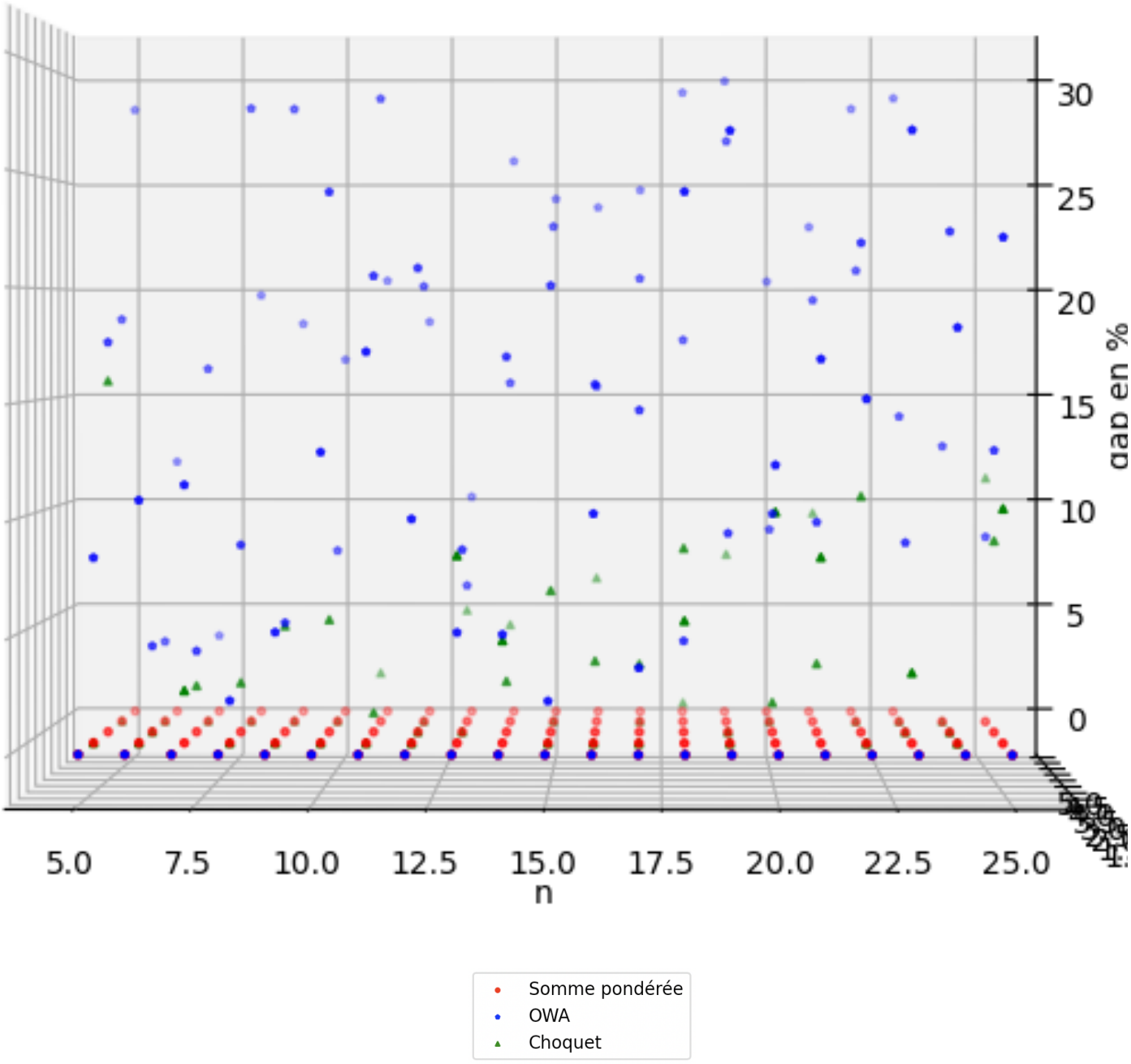
Face à la solution optimale de l'instance du problème calculé grâce à la résolution par programmation linéaire :



Pour la somme pondérée, on voit que le gap est toujours nul, ce qui veut dire que la solution optimale pour un décideur dont les préférences sont représentées par une somme pondérée est bien présente dans les non dominés calculés par PLS. Cela peut s'expliquer que la solution optimale dans ce cas là est supportée donc possiblement plus facile à "trouver" pour PLS.

Pour p=1, le gap est aussi toujours nul, ce qui est normal car on revient à du mono-objectif.

OWA a les plus grands gaps, augmentant de façon très légère avec p. Cela veut dire que PLS ne trouve pas forcément les OWA-meilleures solutions, ou que l'élicitation est moins efficace avec OWA. En fait, nous allons voir que OWA pose en effet beaucoup moins de questions que les deux autres agrégateurs, peut-être y a-t-il une corrélation ?



Voici un autre angle du graphe, nous pouvons voir que Choquet augmente avec la valeur de n mais cette valeur n'affecte pas OWA.

3.1.4. Statistiques

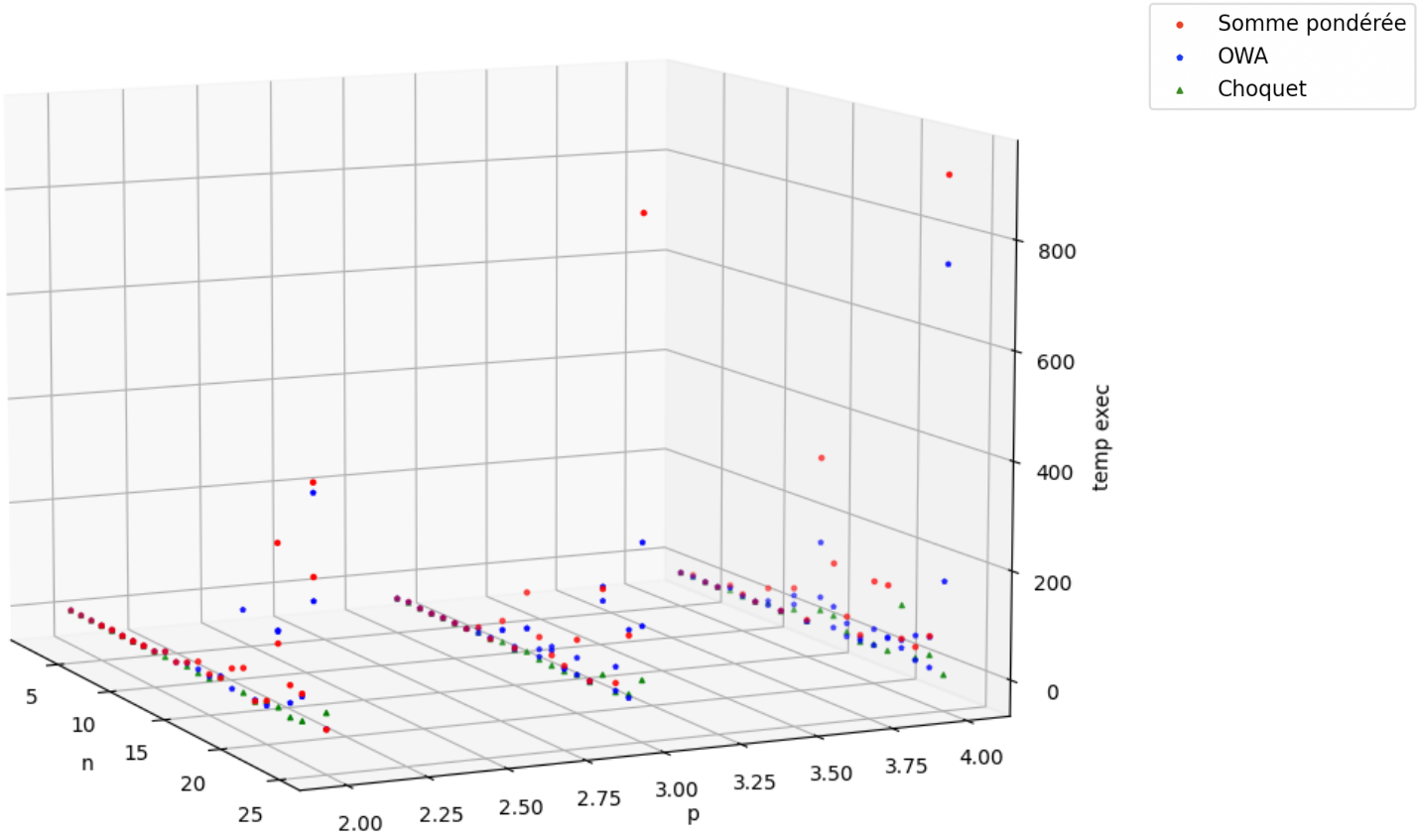
Voici quelques statistiques sur l'élicitation incrémentale avec la somme pondérée, OWA et Choquet. Ceux-ci ont été calculés en moyennant 20 élicitations (20 pour chaque agrégateur) avec une instance pour n=20 et p=4.

Chiffres arrondi au centième :

	Somme pondérée	OWA	Choquet
Temps d'exécution moyen (s)	133,86	37,76	19,9
Nombre de questions moyen	3,25	1	2,35
Gap moyen	0	0	1,42
Gap "réel" moyen	0	23,63	5,66

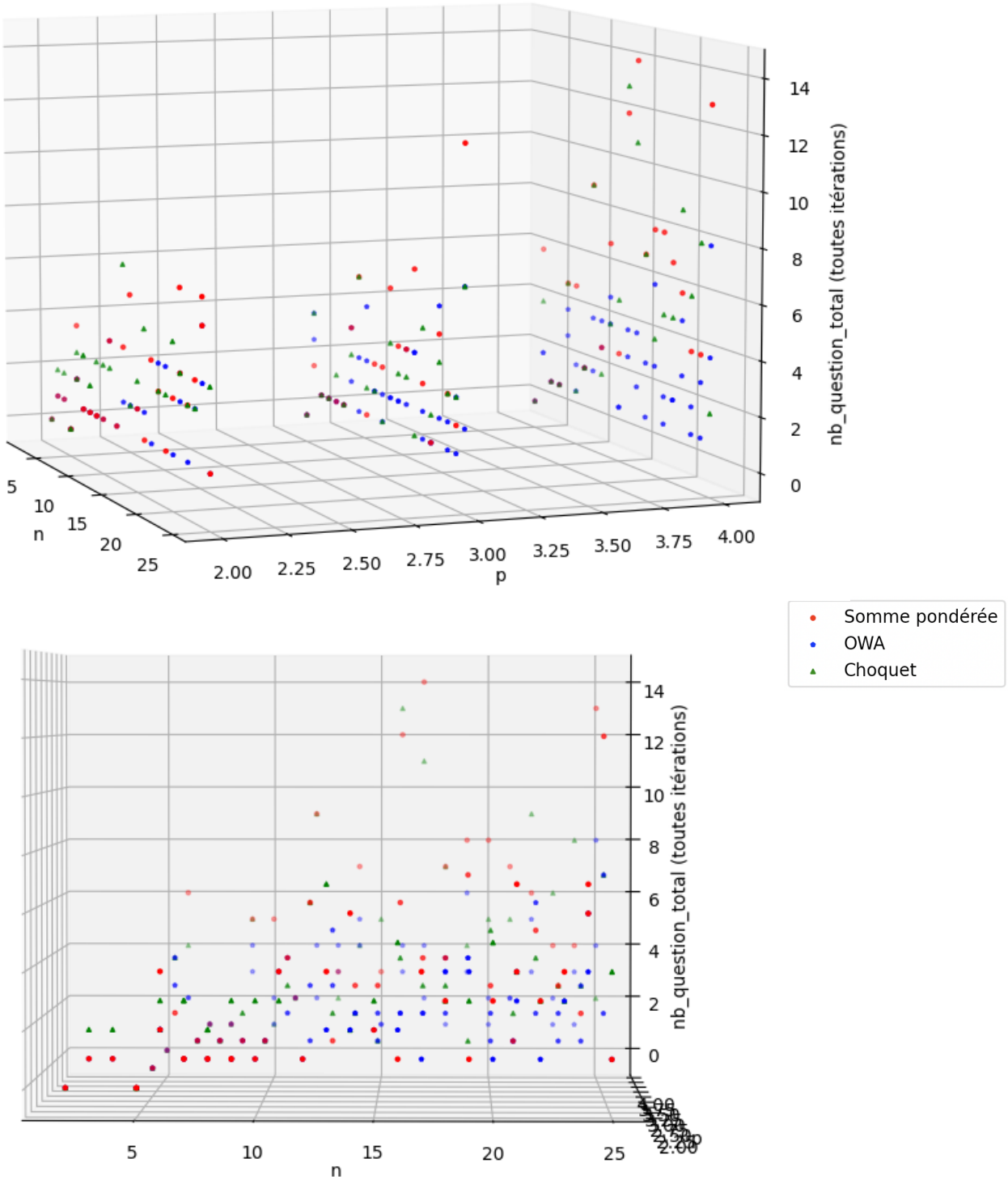
3.2. Deuxième procédure de résolution

3.2.1. Temps d'exécution



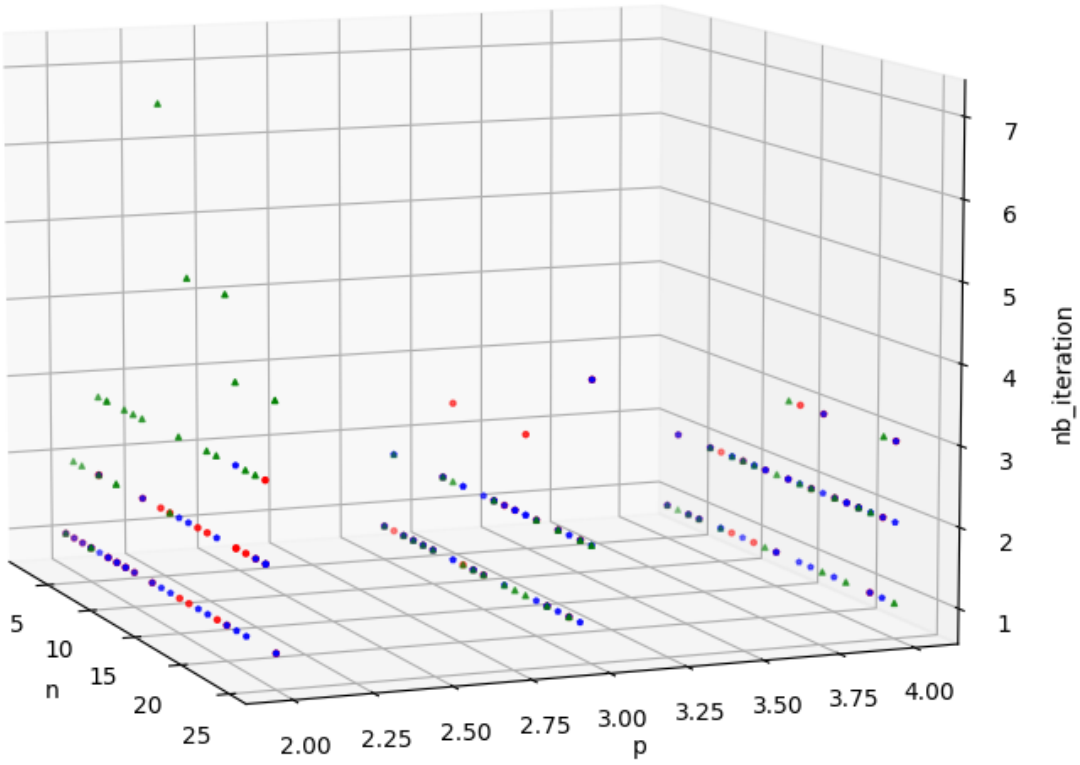
Les temps d'exécution des sommes pondérées semblent être les plus grands, suivis de ceux de OWA puis enfin de ceux de Choquet. Nous voyons clairement que les temps d'exécution dépendent plus de n que de p.

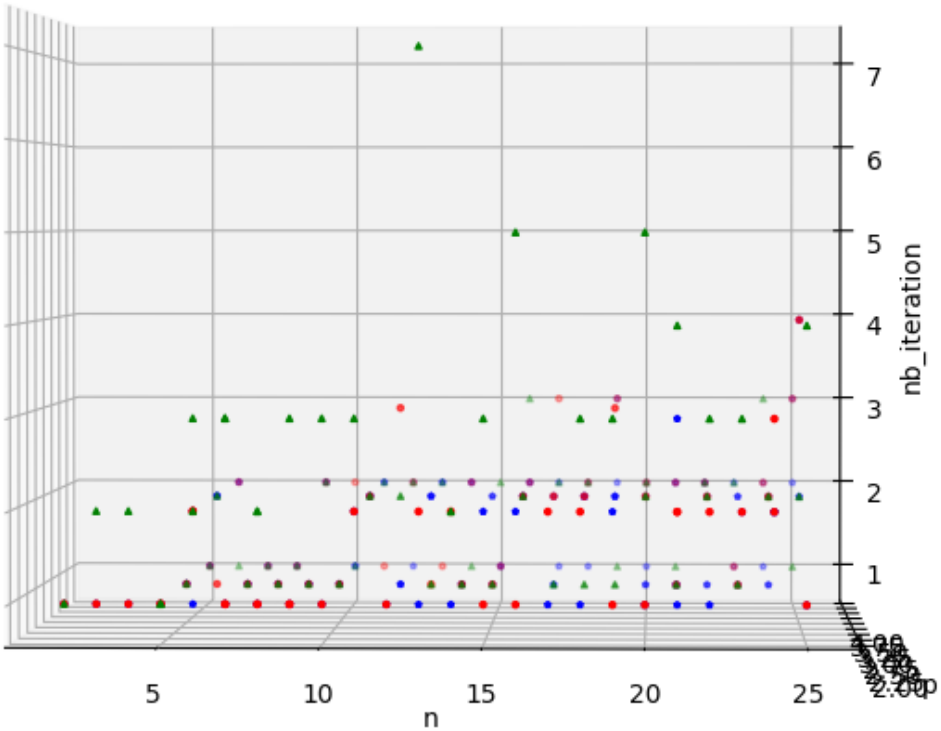
3.2.2. Nombre de question posées



Ces deux graphes sont les mêmes à une rotation près. On peut voir que le nombre de questions augmente avec n ou p (il suffit de fixer par exemple p et de regarder l'évolution de n) mais semble augmenter plus vite avec n. OWA semble poser le moins de questions, tandis que la somme pondérée a une grande variance. Choquet quant à lui se pose de façon consistante plus de questions que OWA.

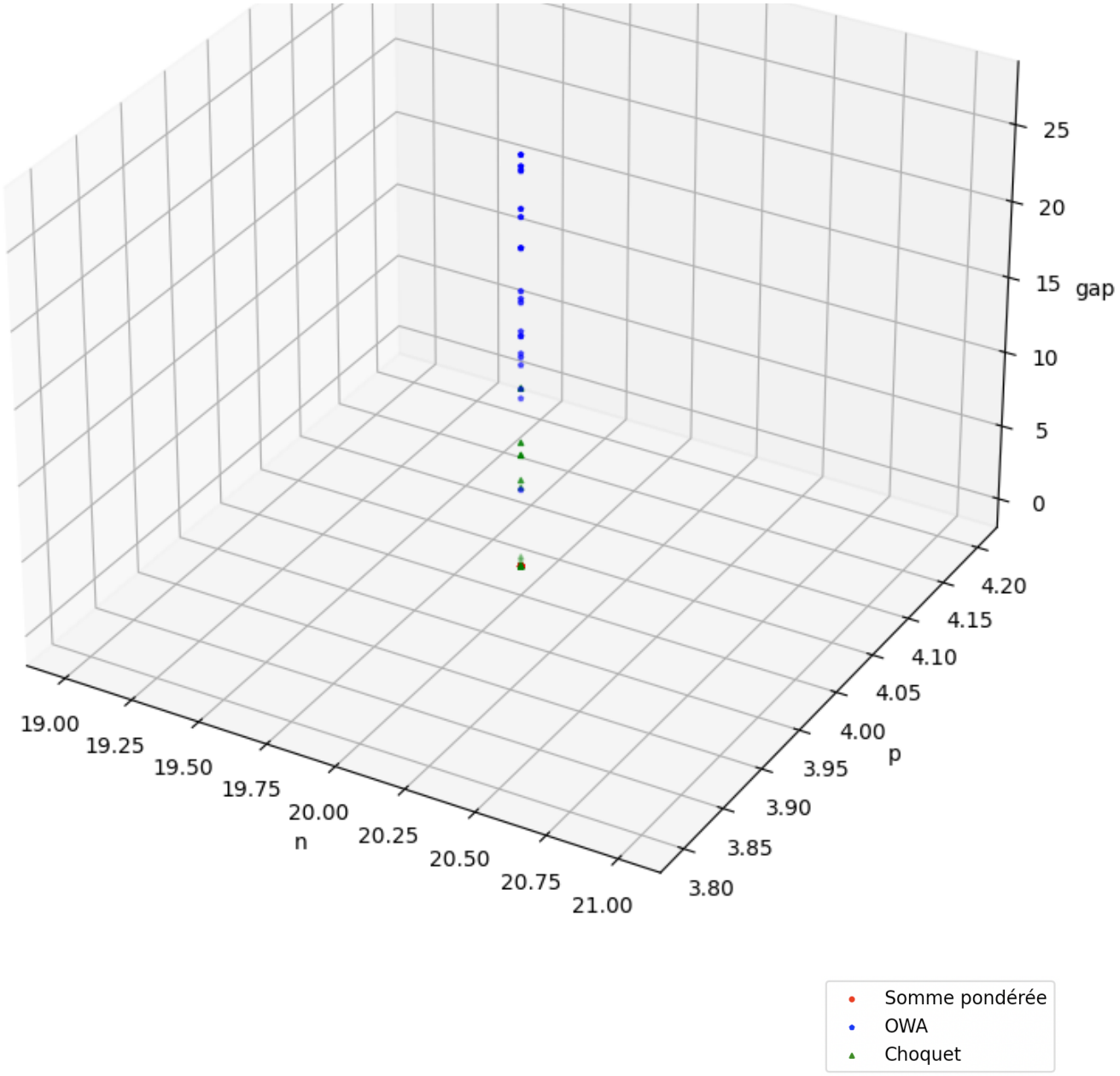
3.2.3. Nombre d'itérations





Ces deux graphes représentent le même graphe à une rotation près. Le nombre d'itérations (nombre de fois qu'on lance une élicitation) ne semble pas être dépendant de n ni de p. À première vue, il semblerait que Choquet soit plus sensible aux double objectifs mais c'est juste que nous avions beaucoup de logs pour Choquet à 2 objectifs, ce qui augmente artificiellement sa présence dans le graphe à cet endroit là. Toutefois, Choquet reste l'agrégateur qui a besoin d'un plus grand nombre d'itération.

3.2.4. Gap



Ce graphe représente le gap par rapport à la vraie solution optimale de l'instance (calculé à partir d'un PL) sur 20 exécutions sur la même instance de OWA, somme pondérée et Choquet chacune et avec des décideurs différents à chaque exécution.

Il est difficile de s'en apercevoir mais tous les points de la somme pondérée ont un gap de 0, ce qui veut dire que la deuxième méthode de résolution est optimale. Choquet présente un gap autour de 5% puis OWA autour de plutôt 15%.

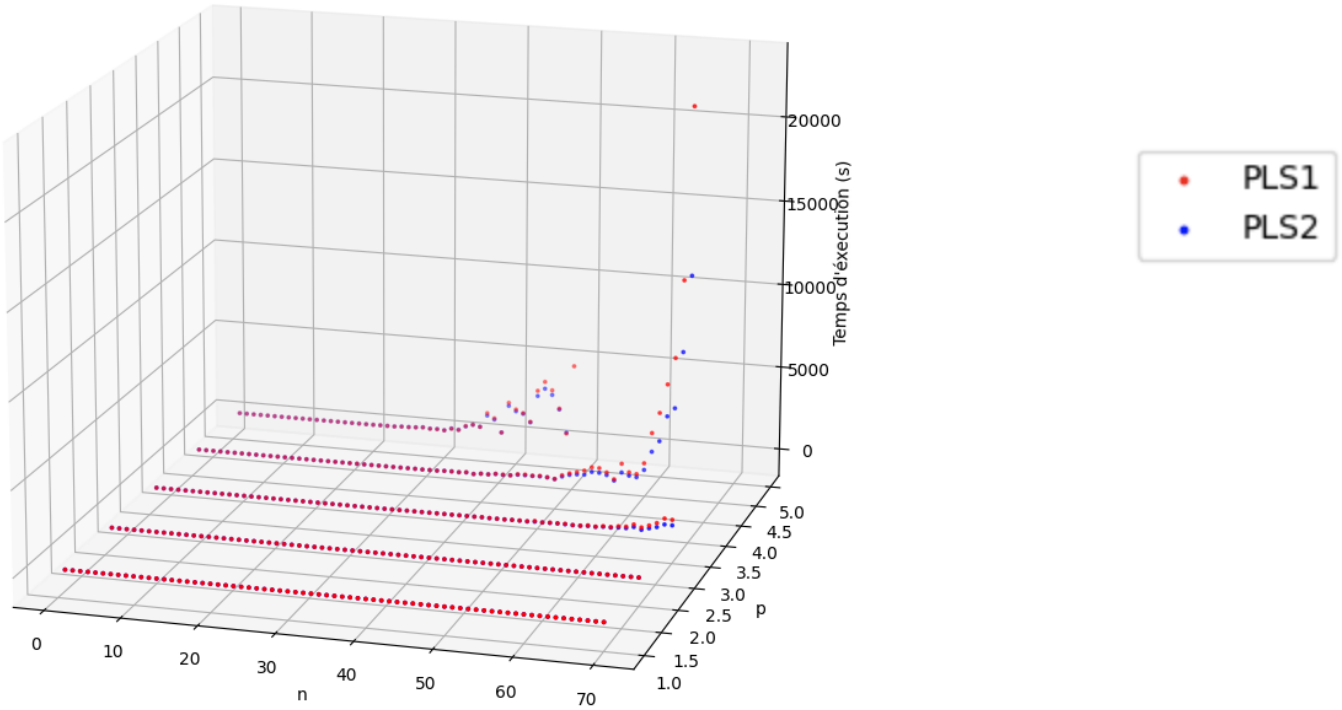
3.2.5. Statistiques

Voici quelques statistiques sur l'élicitation avec recherche locale ayant pour agrégateur la somme pondérée, OWA et Choquet. Ceux-ci ont été calculés en moyennant 20 élicitations (20 pour chaque agrégateur) avec une instance de paramètres n=20 et p=4.

Arrondi au centième :

	Somme pondérée	OWA	Choquet
Temps d'exécution moyen (s)	194,49	44,49	30,90
Nombre de questions moyen	7,5	2	5,7
Gap "réel" moyen	0	17,21	0,98

3.3. PLS



Voici un graphe détaillant les temps d'exécutions pour les deux variantes de PLS. Comme prévu, PLS2, qui implémente une façon plus intelligente de gérer les ensembles non dominés, est plus rapide. On voit aussi que c'est le nombre de critère p qui est le facteur limitant dans l'exécution.

3.4. Comparaison entre la première et deuxième procédure de résolution

Voici le tableau qui résume le changement entre la première procédure et la deuxième. Par exemple, la deuxième procédure met 45% plus de temps en moyenne pour faire l'élicitation que la première sur l'agrégateur somme pondérée.

	Somme pondérée	OWA	Choquet
Temps d'exécution moyen	+45%	+18%	+55%
Nombre de questions moyen	+130%	+100%	+142%
Gap "réel" moyen	+0%	+37%	-83%

On en déduit que pour la somme pondérée et OWA, il est préférable d'utiliser la première méthode. Pour Choquet, la deuxième méthode à des résultats plus proches de l'optimum réel mais prend plus de temps et pose plus de questions.