Kawthar Alaoui Mhammedi

Thibault DUCRUET

Rapport Optimisation discrète
Projet Knapsack

Introduction

Pour répondre aux attentes de ce projet, nous avons implémenté trois algorithmes de métaheuristique (tabou, recuit simulé et génétique), ainsi qu'une résolution linéaire de ce problème. Tout le code peut être retrouvé dans le fichier projet.ipynb avec des commentaires pour aider à la compréhension des fonctions et de comment les tester. Dans ce rapport, nous allons, dans un premier temps, comparer le temps d'exécution des différents algorithmes. Ensuite, comparer leur résultat avec les solutions optimales trouvées grâce à la résolution linéaire. Et enfin, comparer les différentes valeurs de paramètres dans chaque algorithme pour trouver les valeurs qui donnent de meilleures solutions.

I - Temps d'exécution

Pour réaliser des tests sur le temps d'exécution des algorithmes, nous avons définis des paramètres par défaut :

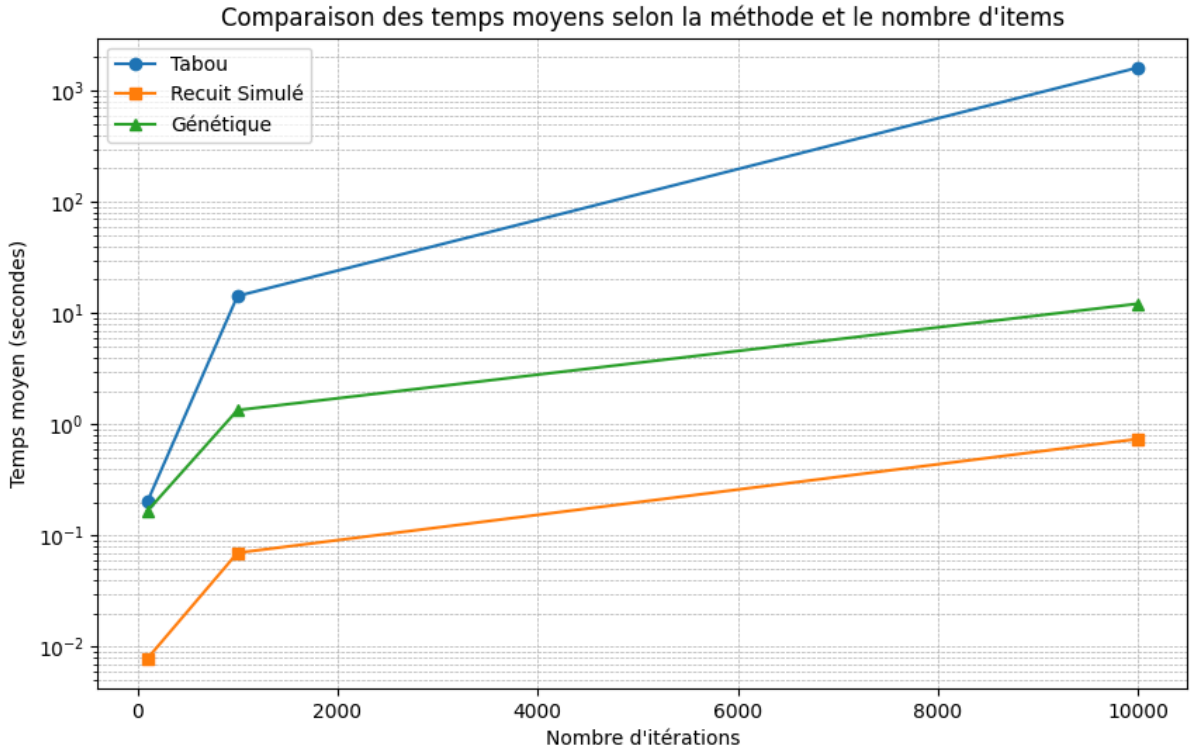
- Tabou : Itérations 500 et Liste Tabou de longueur 1
- Recuit simulé : Température initial 100, Facteur de refroidissement 0.99, Température minimale 0.1 et Itérations max 1000
- Génétique : Probabilité croisement 0.5, Reproduction stratégie élitiste 20, Nombre génération 100 et Nombre individue 100

Voici les résultats des temps d'exécution des différents algorithmes:

Fichier de donnée	Tabou	Recuit Simulé	Génétique
12.100	204 ms ± 9.45 ms	7.67 ms ± 207 µs	150 ms ± 3.48 ms
12.1000	14 s ± 1.49 s	70.9 ms ± 834 µs	1.2 s ± 18.9 ms
12.10000	28 min 2s ± 2 min	736 ms ± 13.1 ms	12 s ± 156 ms
13.100	214 ms ± 14.1 ms	7.99 ms ± 148 µs	177 ms ± 4.11 ms
13.1000	13.6 s ± 92.4 ms	68.7 ms ± 1.11 ms	1.51 s ± 110 ms
13.10000	28 min 12s ± 2 min	745 ms ± 18.9 ms	12.4 s ± 208 ms

15.100	189 ms ± 2.77 ms	7.89 ms ± 72.4 µs	173 ms ± 10 ms
15.1000	15.2 s ± 57.9 ms	70.4 ms ± 1.67 ms	1.32 s ± 44.6 ms
15.10000	24 min 32s ± 2 min	737 ms ± 6.19 ms	12.1 s ± 234 ms

Pour mettre ces résultats dans un graphique, nous avons fait une moyenne des temps d'exécution en fonction du nombre d'items (100, 1000 et 10 000) :



Comme on peut le voir sur ce graphique, la méthode du tabou prend énormément de temps, plus il y a de données, allant jusqu'à plus de 103 secondes, donc plus de 16 minutes. Quant à la méthode génétique, elle reste plutôt rapide, même avec 10000 données ne prenant qu'une dizaine de secondes pour terminer. Et enfin, l'algorithme le plus rapide est le recuit simulé qui reste sous une seconde pour les trois valeurs testées. La lenteur de la méthode Tabou vient certainement du fait de recalculer des voisins à chaque itération, ce qui a de grandes conséquences sur son temps d'exécution et les données enregistrées. Les deux autres méthodes utilisent quant à elles de l'aléatoire pour éviter de calculer autant de voisins d'un coup.

II - Qualité des solutions obtenues

Pour vérifier la qualité des différentes solutions, nous avons dû implémenter une résolution linéaire pour trouver la solution optimale de chaque fichier de données, pour les comparer et mesurer le pourcentage de qualité des réponses fourni par nos algorithmes. Voici donc ces résultats optimaux :

Fichier de donnée	Résultat optimale
12.100	970
12.1000	4514
12.10000	45105
13.100	1989
13.1000	6513
13.10000	64077
15.100	1011
15.1000	4950
15.10000	50622

Pour réaliser ces tests, nous avons pris les mêmes paramètres par défaut que pour le test de vitesse qui sont :

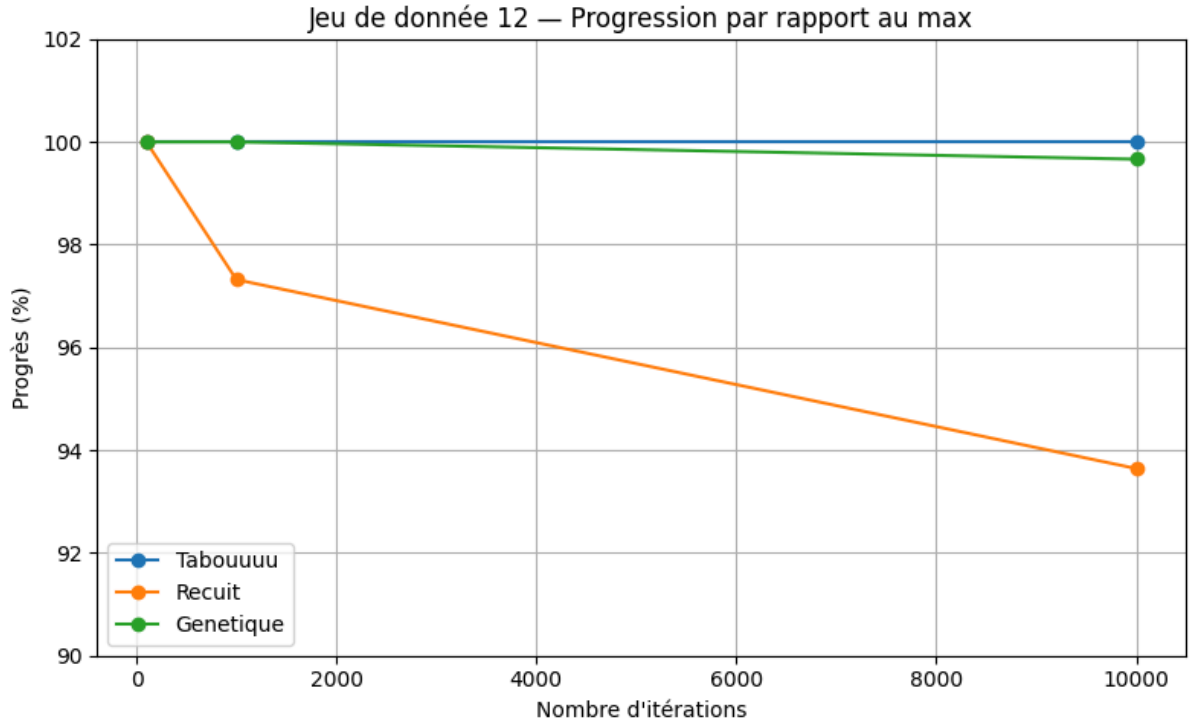
- Tabou : Itérations 500 et Liste Tabou de longueur 1
- Recuit simulé : Température initial 100, Facteur de refroidissement 0.99, Température minimale 0.1 et Itérations max 1000
- Génétique : Probabilité croisement 0.5, Reproduction stratégie élitiste 20, Nombre génération 100 et Nombre individue 100

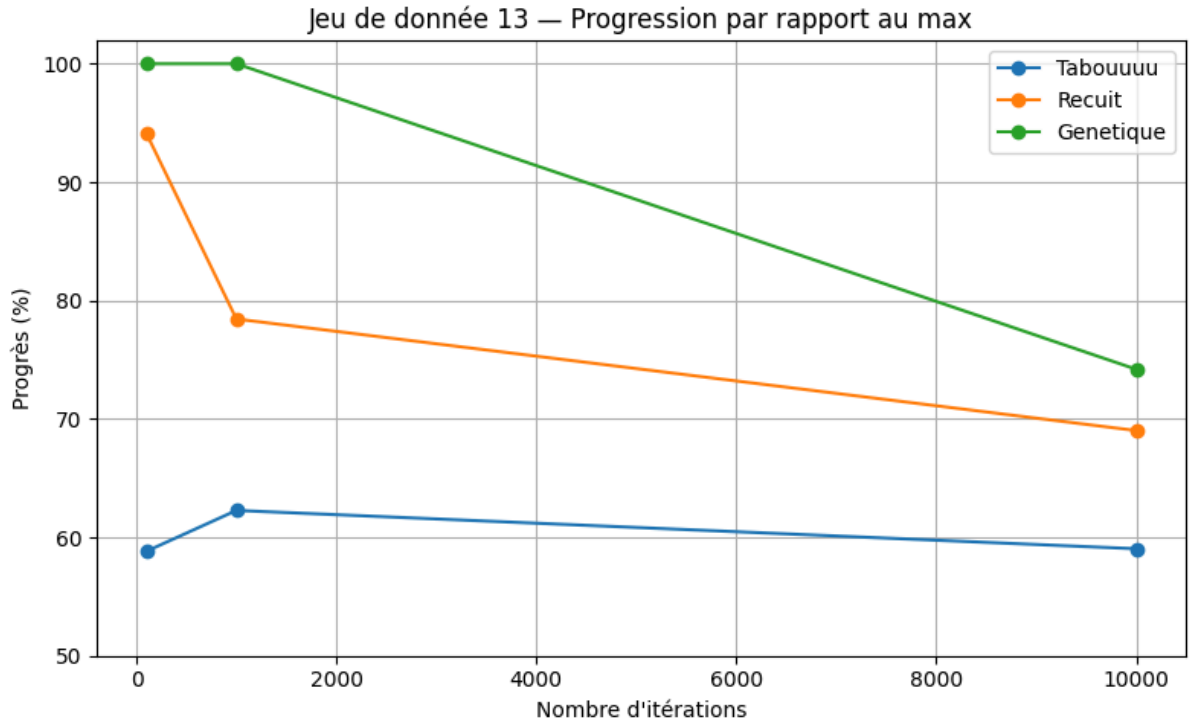
De plus, pour le recuit simulé et la génétique, les résultats choisis sont les meilleurs trouvés après une dizaine d'exécutions, étant donné que leur valeur peut changer avec leur aléatoire. Voici le tableau regroupant ces résultats et leur pourcentage de qualité par rapport à la solution optimale :

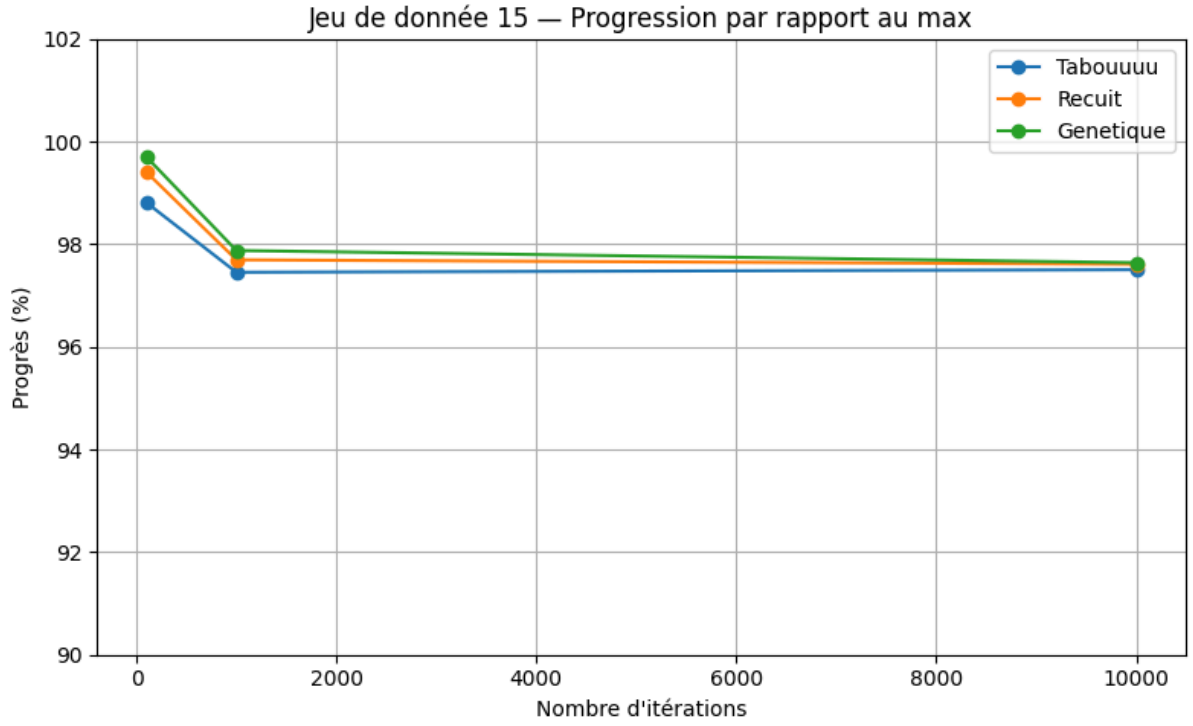
Fichier de donnée	Tabou (val & %)	Recuit simulé (val & %)	Génétique (val & %)
12.100	970 (100.00%)	970 (100.00%)	970 (100.00%)
12.1000	4514 (100.00%)	4393 (97.32%)	4514 (100.00%)

12.10000	45105 (100.00%)	42237 (93.63%)	44953 (99.66%)
13.100	1170 (58.81%)	1872 (94.11%)	1989 (100.00%)
13.1000	4056 (62.29%)	5109 (78.45%)	6513 (100.00%)
13.10000	37830 (59.04%)	44226 (69.01%)	47541 (74.19%)
15.100	999 (98.81%)	1005 (99.41%)	1008 (99.71%)
15.1000	4824 (97.45%)	4836 (97.70%)	4845 (97.88%)
15.10000	49359 (97.51%)	49413 (97.61%)	49428 (97.64%)

Avec ces données, nous allons pouvoir créer trois graphiques représentant les trois jeux de donnée différents (12, 13 et 15) :







D'après ces graphiques, on peut voir dans un premier temps que la manière de génération des données influe énormément sur la capacité des algorithmes à trouver une solution de qualité. En effet, pour les données 12 et 15, les solutions trouvées sont au-dessus de 90% mais elles descendent sous les 75% pour les données 13. Ensuite, concernant les algorithmes, on peut voir que la génétique apporte généralement de très bon résultats, mieux que les autres pour toutes les générations

de données. Ce qui n'est pas le cas du tabou étant donné que pour les données 13, il reste vraiment très peu précis restant autour des 60%. Enfin, le recuit simulé à l'air de baisser en qualité plus il y a de données, peu importe leur génération. On peut donc en conclure que le tabou est une bonne solution uniquement pour certaines génération de données précises, sinon, il vaut mieux privilégier la génétique qui est plus versatile et donne dans tous les cas d'assez bon résultat.

III - Changement des paramètres

1 - Recuit Simulé

a. Influence de la température initiale T

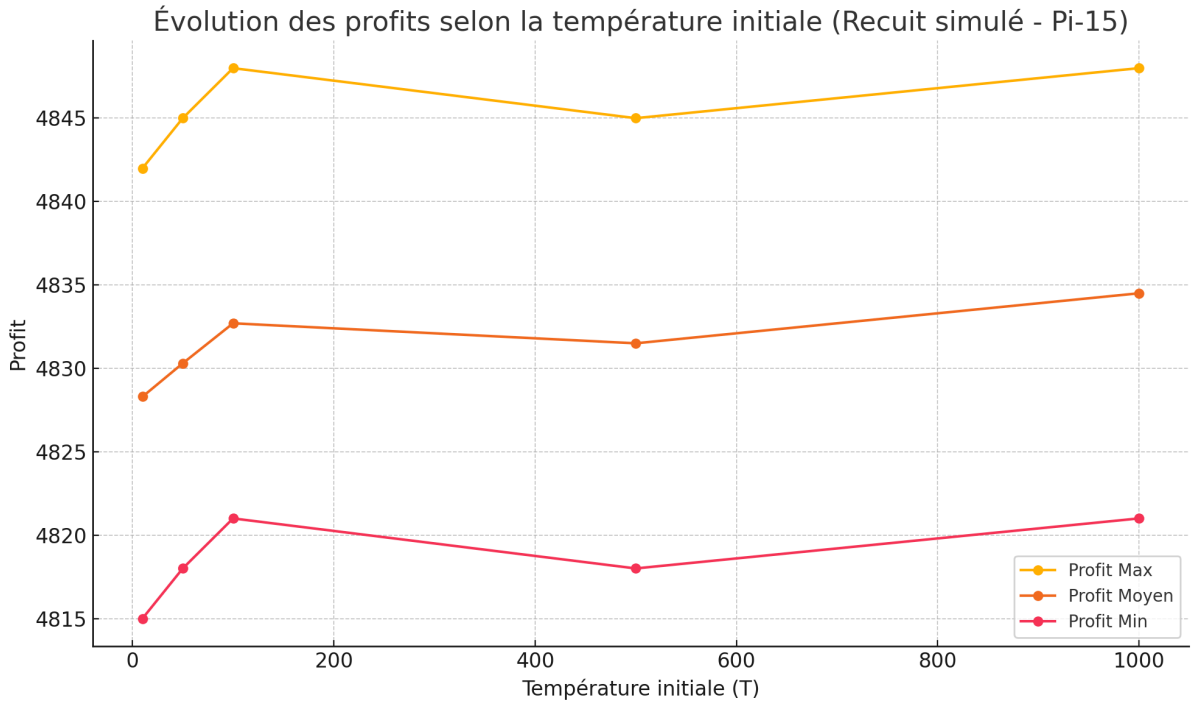
Dans cette série d'expérimentations, nous avons fixé les paramètres suivant pour l'algorithme de recuit simulé:

- **alpha = 0.99 (facteur de refroidissement)**
- **Tmin = 0.1 (température minimale)**
- **iterMax = 1000 (nombre maximal d'itérations)**

Nous avons fait varier la température initiale T parmi les valeurs suivantes : 10, 50, 100, 500, 1000. Pour chaque valeur de T, nous avons exécuté l'algorithme 10 fois. Cette analyse a été menée sur les trois jeux de données Pi-15, Pi-13 et Pi-12.

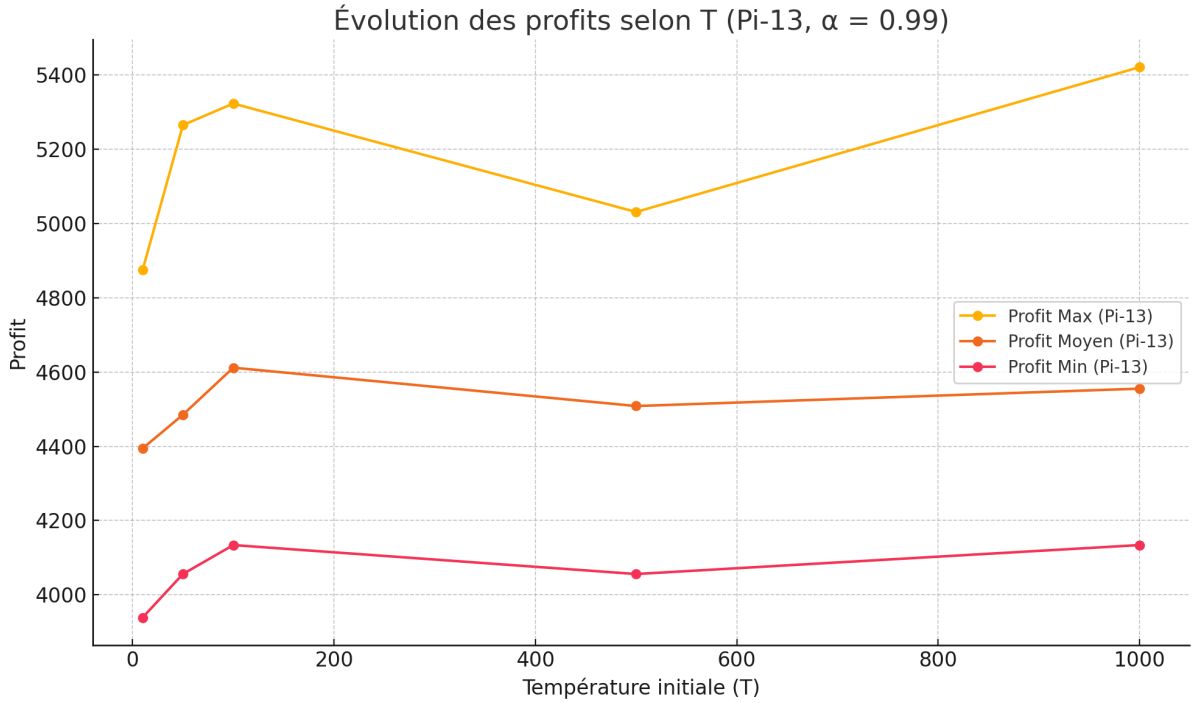
Pi-15

T	Diffs profits repérés	Profit Max	Profit Min	Profit Moyen	Écart type
10	4830,4827,4836,4816,4836,4842,4815,4818,4839,4824	4842	4815	4828.3	9.87
50	4830,4821,4836,4830,4827,4824,4839,4818,4833,4845	4845	4818	4830.3	8.29
100	4830,4839,4827,4824,4836,4827,4842,4833,4821,4848	4848	4821	4832,7	8.54
500	4827,4830,4824,4833,4839,4818,4842,4836,4845,4821	4845	4818	4831.5	9.08
1000	4830,4836,4833,4827,4842,4839,4824,4845,4848,4821	4848	4821	4834.5	9.08



Pi-13

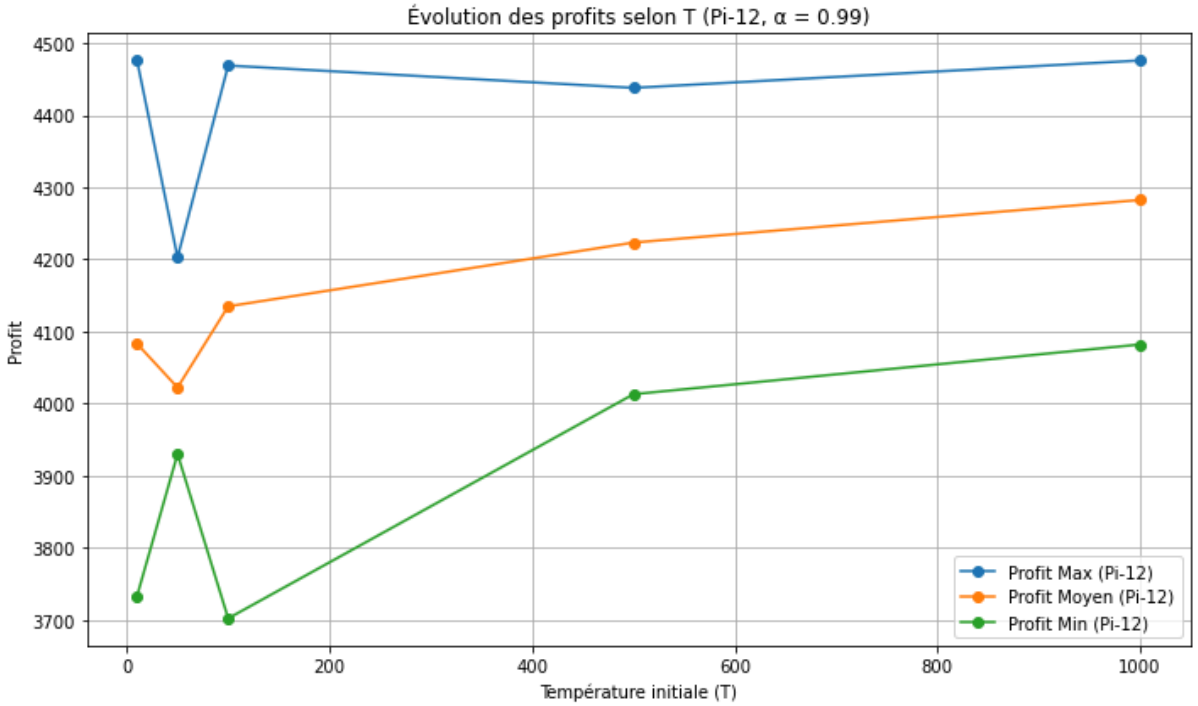
T	Diffs Profits repérés	Profit max	Profit min	Profit moyen	Écart type
10	4875, 4212, 4680, 4602, 4290, 3939, 4368, 4446, 4134, 4797	4875	3939	4394.3	289.92
50	4524, 4368, 4446, 4212, 4056, 4875, 4680, 4290, 5265, 4134	5265	4056	4485	351.43
100	4680, 4134, 4797, 4290, 4446, 4602, 4524, 4212,5323,5109	5323	4134	4611.7	363.19
500	4290, 4368, 4524, 4056, 5031, 4212, 4875, 4446, 4602, 4680	5031	4056	4508.4	284.62
1000	4602, 4524, 4680, 4134, 4290, 5421, 4446, 4875, 4368, 4212	5421	4134	4555.2	373.25



Pi-12

T	Diffs Profits repérés	Profit max	Profit Min	Profit Moyen	Écart type
10	4248, 4400, 4120, 4013, 3892, 3809, 4051, 3733, 4476, 4089	4476	3733	4083,1	228,81
50	3930, 4089, 4203, 4165, 3975, 3892, 4013, 3968, 3937, 4051	4203	3930	4022,3	98,66
100	3702, 4393, 4158, 4082, 3847, 4469, 4279, 3892, 4127, 4400	4469	3702	4134,9	245,37
500	4324, 4127, 4158, 4013, 4120, 4089, 4438, 4286, 4362, 4317	4438	4013	4223,4	132,16

1000	4324, 4286, 4127, 4248, 4438, 4082, 4165, 4317, 4476, 4362	4476	4082	4282,5	122,64



Analyse des résultats

Globalement, on observe une tendance commune sur les trois jeux de données:

- Lorsque la température initiale T est faible (T=10 ou 50), les résultats sont souvent moins bons. En effet, le recuit simulé accepte peu de mauvaises solutions, ce qui limite l'exploration de l'espace de recherche et peut conduire à des optima locaux.
- À l'inverse, des valeurs plus élevées de T (500 ou 1000) permettent une meilleure diversification initiale. Cela se traduit, dans certains cas, par une légère amélioration du profit moyen, même si le profit maximum reste souvent identique.

On peut conclure que , même si une température trop faible limite l'exploration de l'espace de recherche, il n'est pas nécessaire de choisir des valeurs très élevées. Un T initial autour de 100 à 500 semble offrir une bonne solution.

b. Influence du facteur de refroidissement (alpha)

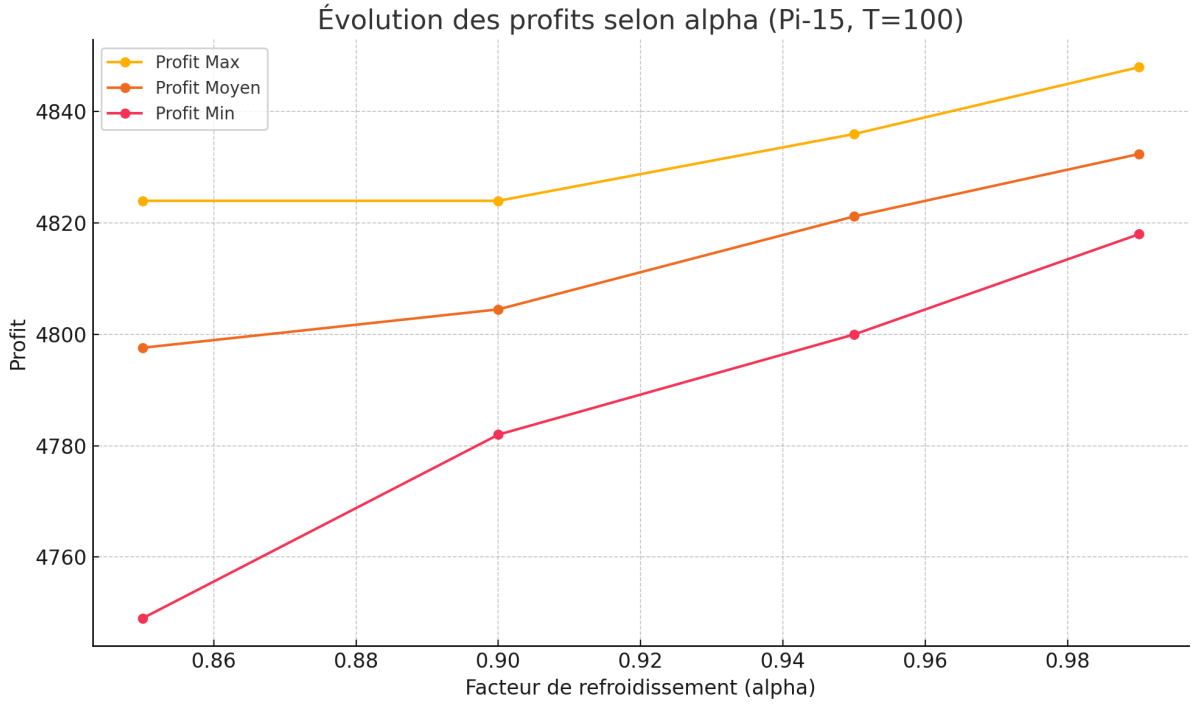
Dans cette série d'expérimentations, nous avons fixé les paramètres suivant pour l'algorithme de recuit simulé:

- **T= 100 (température initiale)**
- **Tmin = 0.1 (température minimale)**
- **iterMax = 1000 (nombre maximal d'itérations)**

Nous avons fait varier le facteur de refroidissement(alpha) parmi les valeurs suivantes : 0.85, 0.9, 0.95, 0.99. Pour chaque valeur de alpha, nous avons exécuté l'algorithme 10 fois. Cette analyse a été menée sur les trois jeux de données Pi-15, Pi-13 et Pi-12.

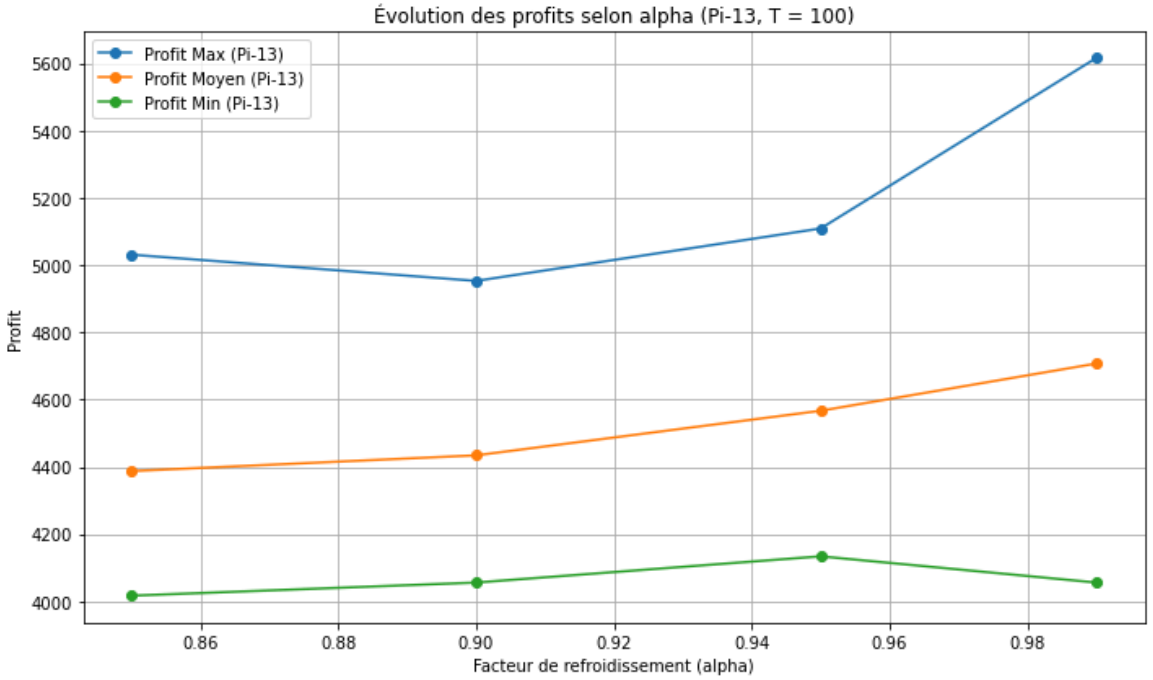
Pi-15

alpha	Diffs Profits repérés	Profit max	Profit min	Profit moyen	Écart type
0,85	4797,4770,4749,4821,4809,4824,4815,4788,4812,4791	4824	4749	4797.6	23.9
0,9	4791,4788,4821,4785,4824,4815,4782,4827,4803,4809	4824	4782	4804.5	17,10
0,95	4812,4830,4836,4833,4821,4818,4800,4827,4806,4809	4836	4800	4821.2	12.44
0,99	4830,4839,4827,4824,4836,4827,4842,4833,4818,4848	4848	4818	4832.4	9.03



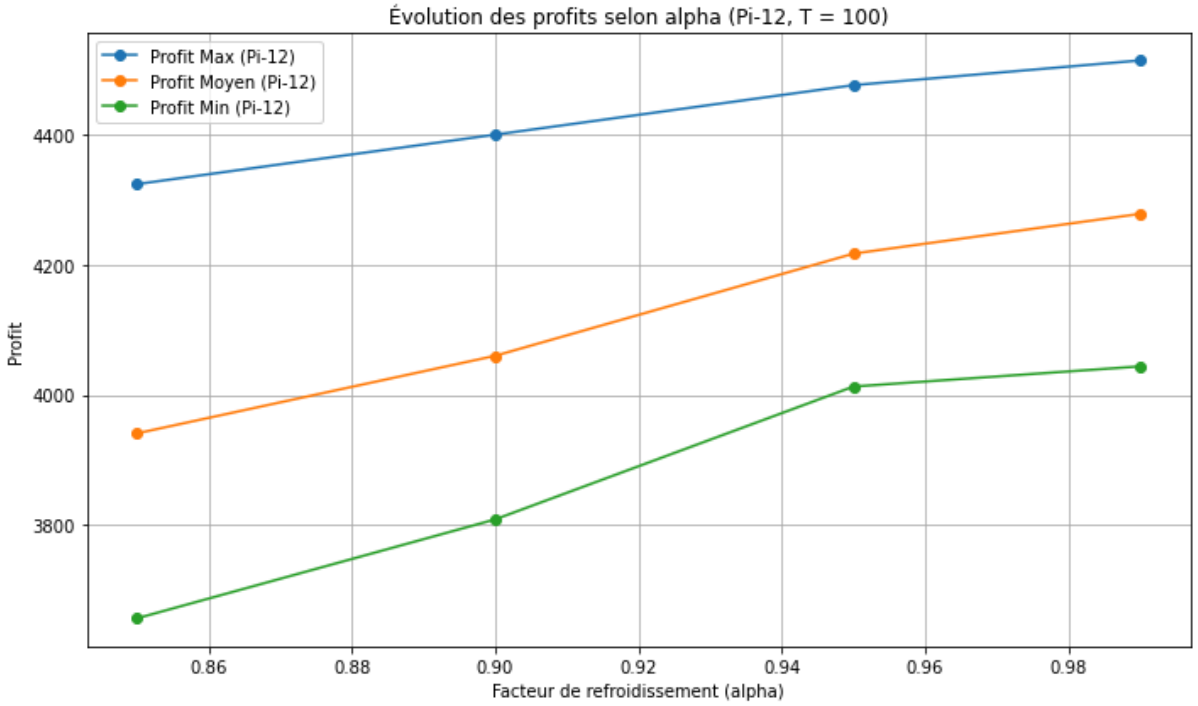
Pi-13

alpha	Diffs Profits repérés	Profit max	Profi t min	Profit moyen	Écart type
0,85	4602, 4212, 4446, 4290, 4368, 4524, 5031, 4017,4251,4134	5031	4017	4387,5	274,43
0,9	4212, 4134, 4290, 4758, 4368, 4602, 4524, 4446,4056, 4953	4953	4056	4434,3	267,97
0,95	4290, 4368, 4446, 4680, 4875, 4524, 4134, 4212, 5031, 5109	5109	4134	4566,9	326,04
0,99	4134, 4524, 4056, 4602,5031, 4680,4290,4953,5616,5187	5616	4056	4707,3	468,35



Pi-12

alpha	Diffs Profits repérés	Profit max	Profit min	Profit Moyen	Écart type
0,85	4324, 4013, 3937, 4127, 4089, 3740, 3816, 3930, 3778, 3657	4324	3657	3941,1	195.77
0,9	4248, 4006, 4089, 4400, 4082, 3816, 4165, 3975, 3809, 4013	4400	3809	4060,3	173,26
0,95	4013, 4089, 4400, 4165, 4051, 4127, 4324, 4248, 4279, 4476	4476	4013	4217,2	146,01
0,99	4317, 4393, 4158, 4082, 4044, 4469, 4279, 4514, 4127, 4400	4514	4044	4278,3	158,84



Analyse globale:

- Nous observons que plus alpha est proche de 1, plus la décroissance de la température est lente, ce qui laisse davantage de temps à l'algorithme pour explorer des solutions de qualité. Cela se traduit généralement par des profits moyens plus élevés et une meilleure stabilité des résultats.
- En revanche, pour des valeurs trop basses d'alpha, la recherche se fige rapidement, ce qui limite l'exploration de nouvelles solutions et entraîne des profits moindres.

Ainsi, un alpha élevé (≥ 0,95) semble offrir le meilleur compromis entre exploration efficace des solutions et convergence vers un optimum.

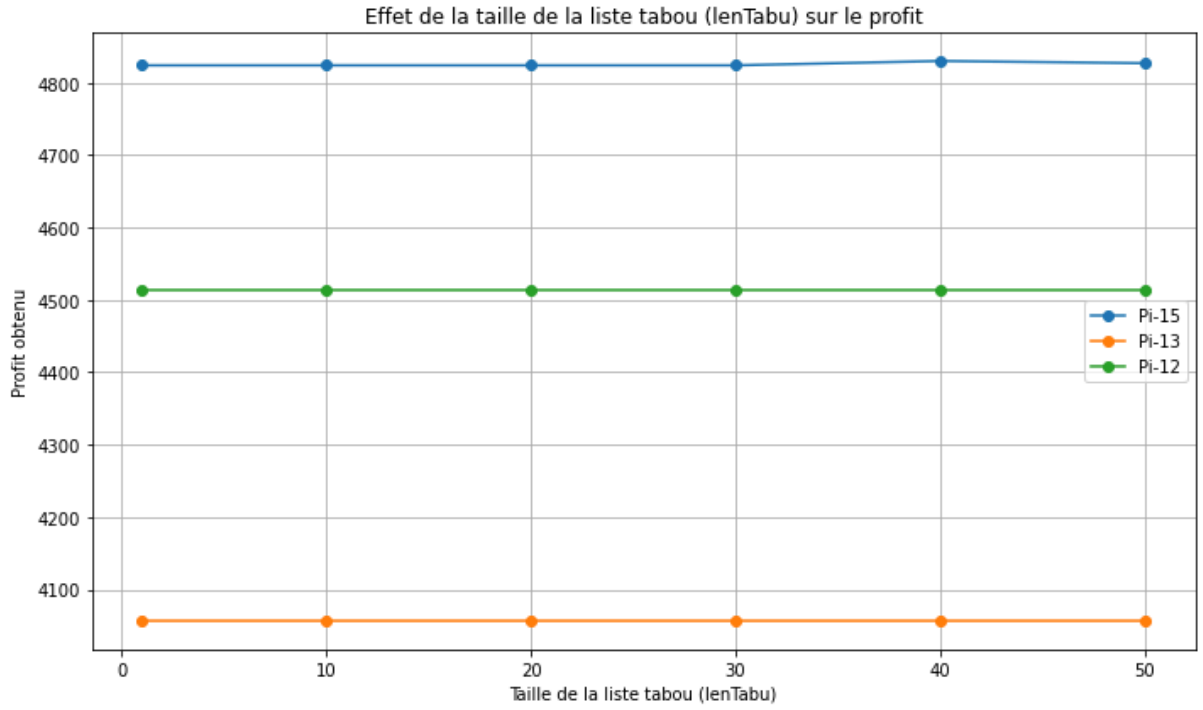
2 - Tabou

Dans cette autre série d'expérimentations, nous avons fixé les paramètres suivant pour l'algorithme génétique:

- maxItération = **500 (nombre d'itération tabou)**

Et nous avons fait varier le paramètre lenTabu représentant le longueur de la liste du tabou entre 1, 10, 20, 30, 40 et 50. Voici donc les solutions pour ces variations selon les différents jeux de données :

Pi-15		Pi-13		Pi-12	
lenTabu	**Profit**	**lenTabu**	**Profit**	**lenTabu**	**Profit**
50	4827	50	4056	50	4514
40	4830	40	4056	40	4514
30	4824	30	4056	30	4514
20	4824	20	4056	20	4514
10	4824	10	4056	10	4514
1	4824	1	4056	1	4514



Premièrement, on voit que pour les données 12, le profit optimal est atteint. Mais concernant les données 13 et 15, l'augmentation de la longueur de la liste tabou n'augmente que très peu ou pas le profit de la solution, alors que la solution trouvée est loin d'être la meilleure. On peut donc en conclure que ces jeux de données ont beaucoup de maximum locaux et donc bloquent le tabou dans sa recherche, peu importe la taille de la liste.

3 - Génétique

a. Influence de la probabilité de croisement probaCross

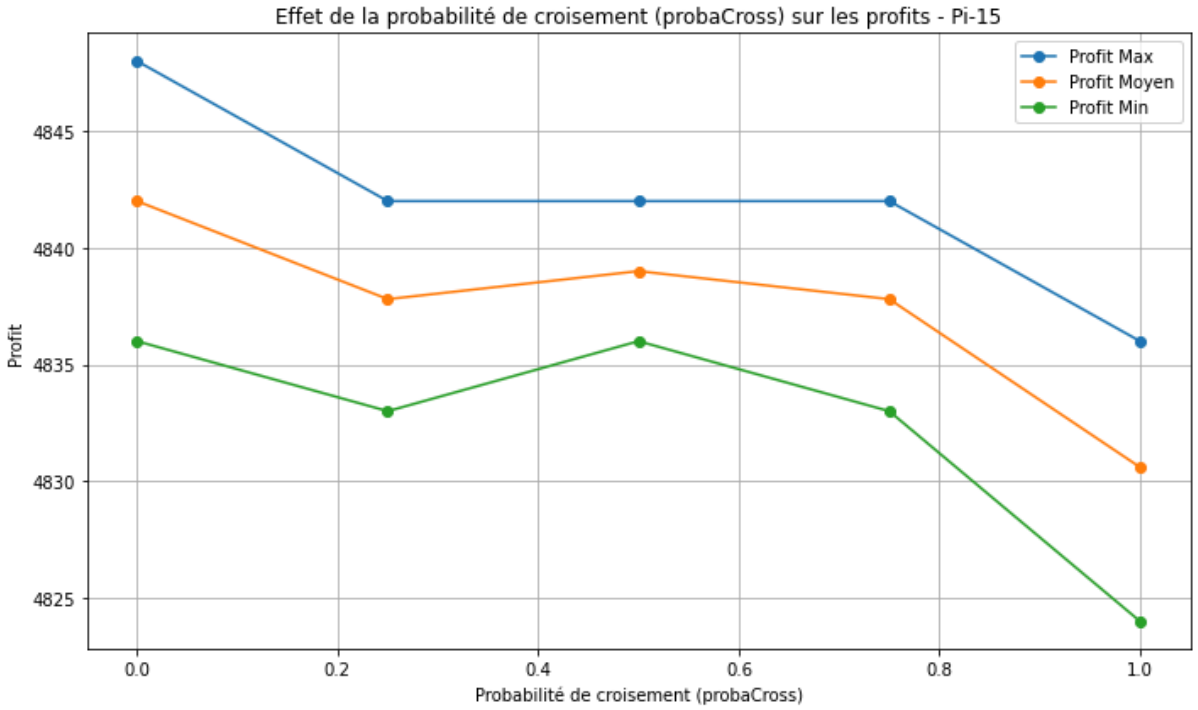
Dans cette série d'expérimentations, nous avons fixé les paramètres suivant pour l'algorithme génétique:

- **nbPop = 100 (nombre d'individu dans la population)**
- **nbGen = 100 (nombre de génération)**
- **nbBest = 20 (nombre d'individu reproduit par élitisme)**

Et nous avons fait varier le paramètre probaCross qui représente la probabilité d'avoir un croisement au lieu d'une mutation entre 0, 0.25, 0.5, 0.75 et 1. Voici donc les solutions pour ces variations selon les différents jeux de données :

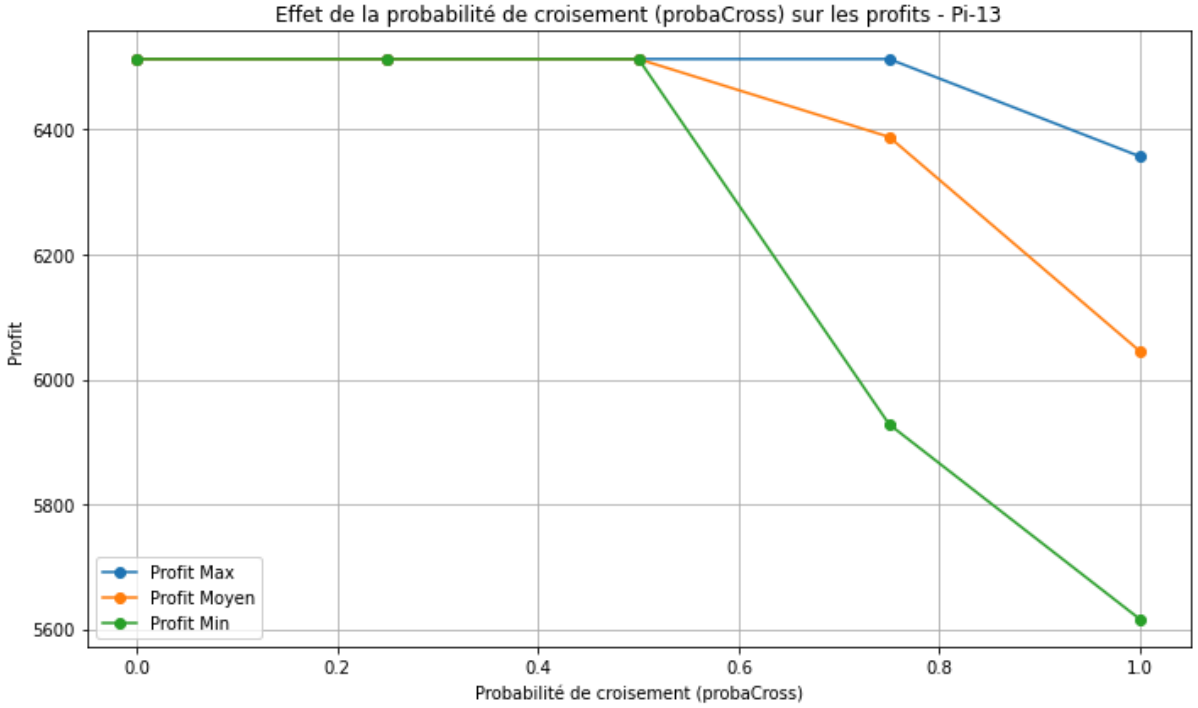
Pi-15

probaCross	Profits	Profit max	Profit min	Profit moyen
0	4839, 4842, 4845, 4836, 4848	4848	4836	4842
0,25	4839, 4833, 4836, 4842, 4839	4842	4833	4837,8
0,5	4839, 4836, 4842, 4836, 4842	4842	4836	4839
0,75	4839, 4839, 4836, 4833, 4842	4842	4833	4837,8
1	4830, 4836, 4833, 4830, 4824	4836	4824	4830,6



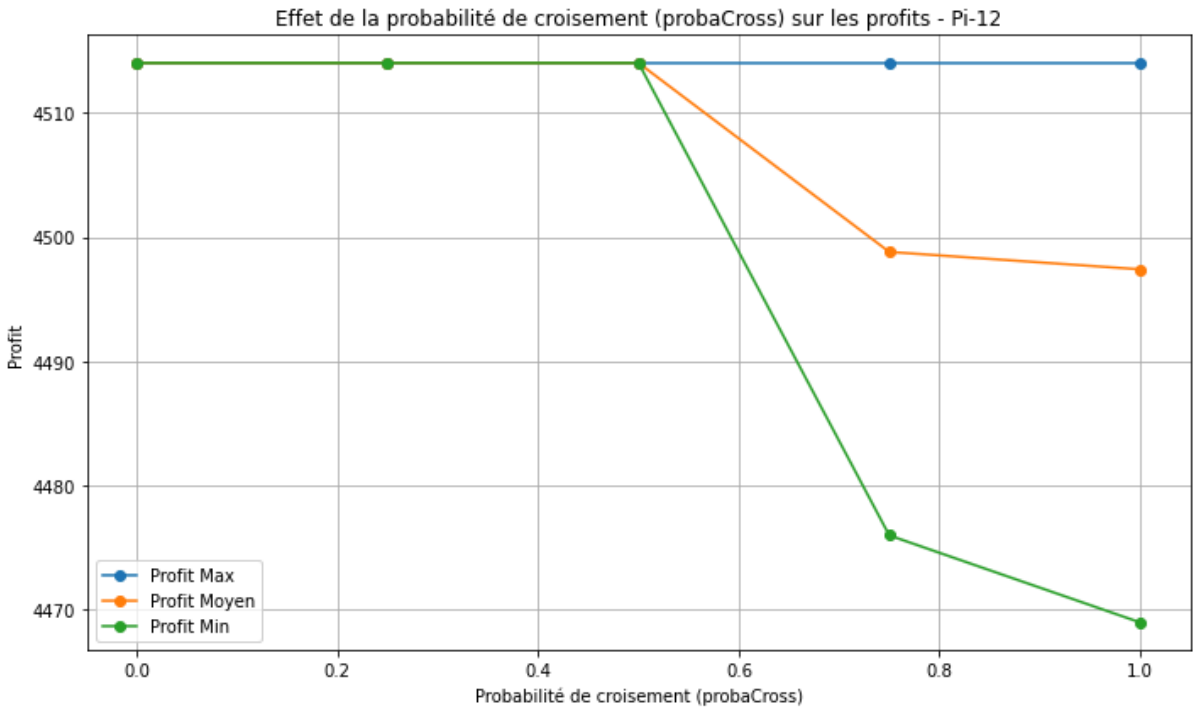
Pi-13

probaCross	Profit	Profit max	Profit min	Profit moyen
0	6513,6513,6513,6513,6513	6513	6513	6513
0,25	6513,6513,6513,6513,6513	6513	6513	6513
0,5	6513,6513,6513,6513,6513	6513	6513	6513
0,75	6513, 5928,6513, 6474,6513	6513	5928	6388,2
1	6084, 5616, 6357, 6123, 6045	6357	5616	6045



Pi-12

probaCross	Profit	Profit max	Profit min	Profit moyen
0	4514	4514	4514	4514
0,25	4514	4514	4514	4514
0,5	4514	4514	4514	4514
0,75	4514,4476,4514,4476,4514	4514	4476	4498,8
1	4514,4476, 4469,4514,4514	4514	4469	4497,4



Analyse des résultats

Grâce à ces graphiques, nous pouvons voir assez clairement que la probabilité de croisement agit bel et bien sur la solution trouvée. En effet, nous voyons que plus la probabilité est faible, plus la solution stagne, voire devient meilleure, et à l'inverse, plus elle augmente, plus il est possible de trouver des résultats moins bons. On peut donc en conclure que dans le cas du problème du sac à dos, faire un croisement de solutions existantes ne bénéficie pas à trouver une solution, cela va plutôt l'empirer. Et donc, que dans le cas de ce problème, les mutations vont souvent apporter de meilleures solutions.

b. Influence du nombre d'individus reproduit par élitisme nbBest

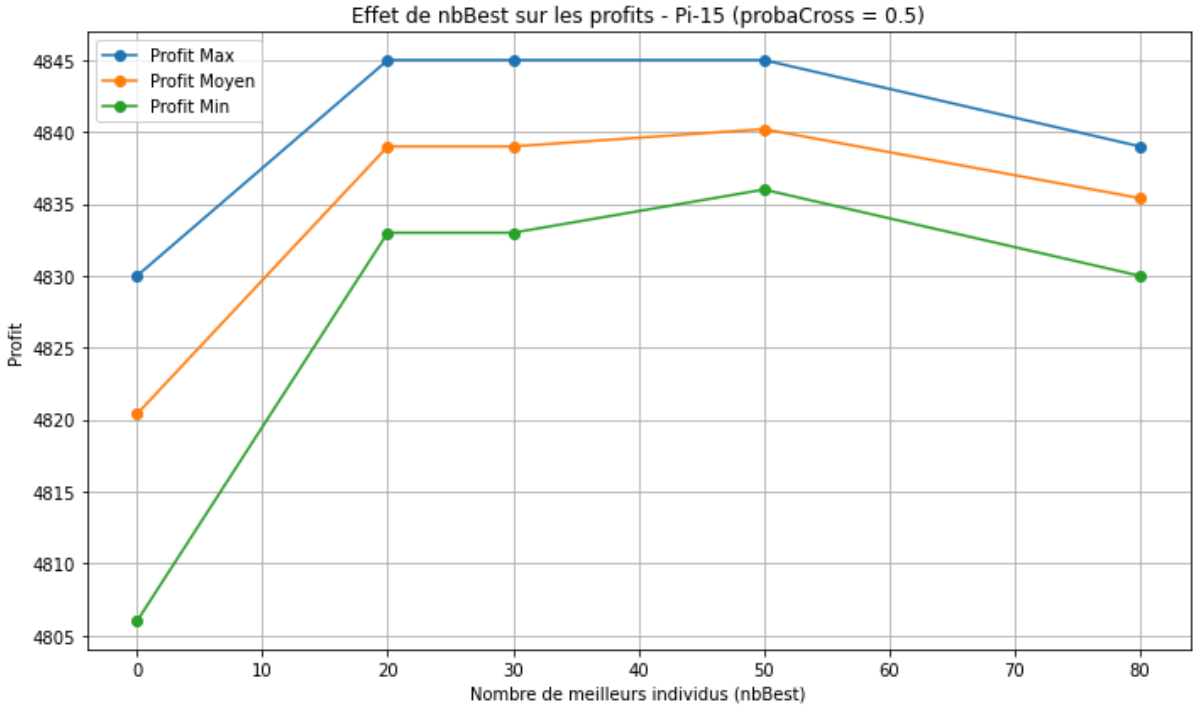
Dans cette autre série d'expérimentations, nous avons fixé les paramètres suivant pour l'algorithme génétique:

- **nbPop = 100 (nombre d'individu dans la population)**
- **nbGen = 100 (nombre de génération)**
- **probaCross = 0.5 (probabilité d'avoir un croisement)**

Et nous avons fait varier le paramètre nbBest représentant le nombre d'individus reproduit par élitisme entre 0, 20, 30, 50 et 80. Voici donc les solutions pour ces variations selon les différents jeux de données :

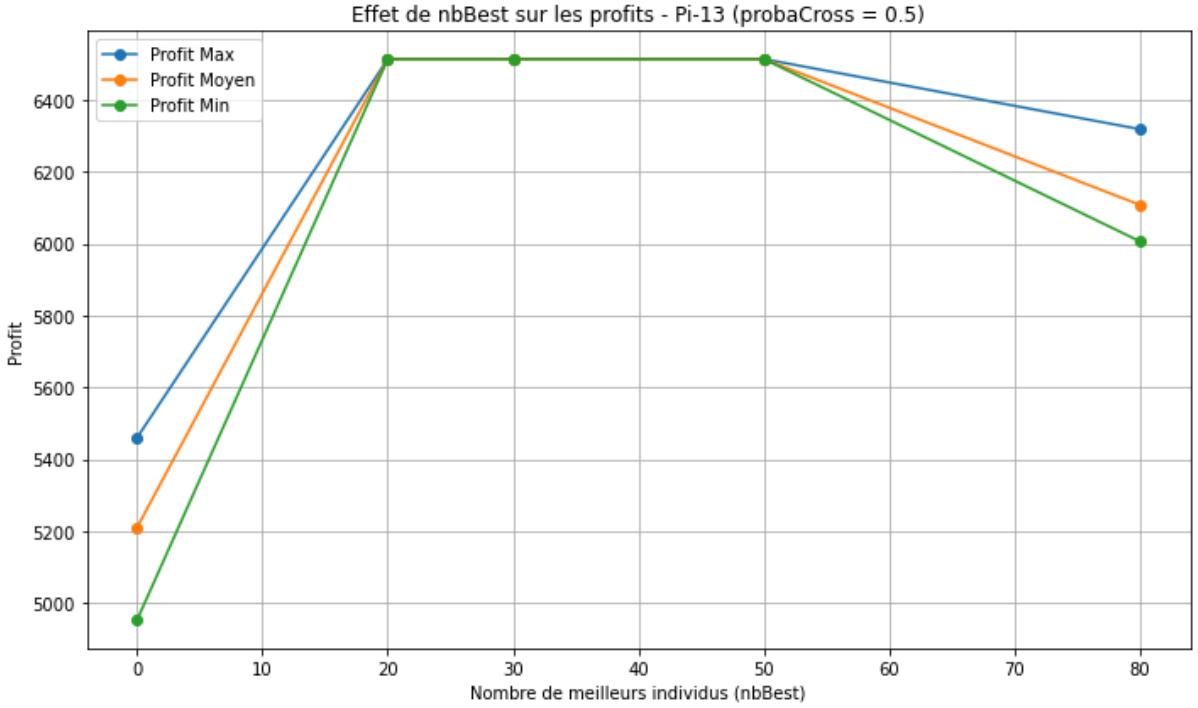
Pi-15

nbBest	Profit	Profit max	Profit min	Profit moyen
0	4824, 4830, 4827, 4806, 4815	4830	4806	4820,4
20	4839, 4833, 4836, 4845, 4842	4845	4833	4839
30	4842, 4836, 4839, 4833, 4845	4845	4833	4839
50	4842, 4836, 4845, 4839, 4839	4845	4836	4840,2
80	4839, 4839, 4830,4836, 4833	4839	4830	4835,4



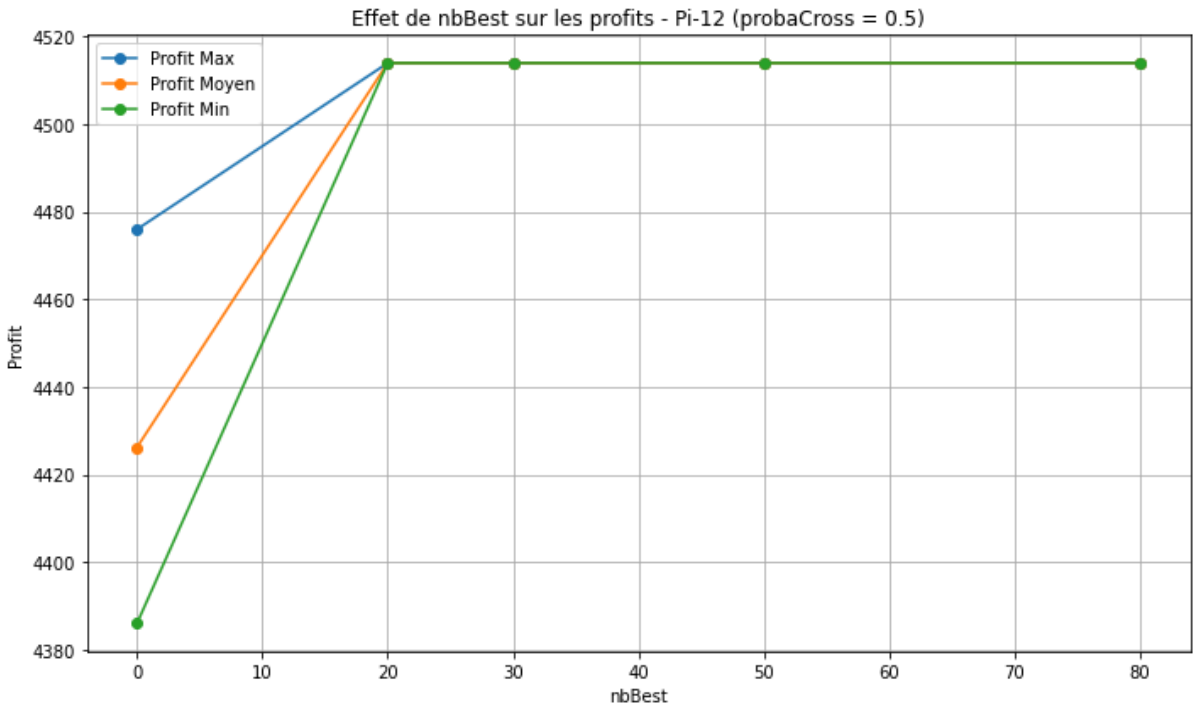
Pi-13

nbBest	Profit	Profit max	Profit min	Profit moyen
0	5265, 5460, 5148, 5226, 4953	5460	4953	5210,4
20	6513	6513	6513	6513
30	6513	6513	6513	6513
50	6513	6513	6513	6513
80	6045, 6318, 5928, 6240, 6006	6318	6006	6107,4



Pi-12

nbBest	Profit	Profit max	Profit min	Profit moyen
0	4393, 4438, 4476, 4438, 4386	4476	4386	4426,2
20	4514	4514	4514	4514
30	4514	4514	4514	4514
50	4514	4514	4514	4514
80	4514	4514	4514	4514



Analyse des résultats

Grâce à ces graphiques, nous pouvons voir assez clairement que le taux de reproduction élitiste agit aussi sur la solution trouvée. En effet, nous voyons que dans la plupart des jeux de données, un taux de 20 à 50 pour cent de cette reproduction permet de maximiser la solution trouvée. De plus, l'utilisation d'un taux aux extrémités, soit 0 ou 80 pour cent, va dégrader la solution.

Nous pouvons donc en conclure que ce paramètre est important dans la recherche d'une bonne solution, étant donné qu'il va permettre de sauvegarder la meilleure partie de la population et de la réutiliser par la suite. Mais il faut tout de même trouver un bon compromis, car en garder trop ne permet pas de trouver de meilleures solutions assez vite avec un croisement ou une mutation.

Conclusion

Ce projet nous a permis d'étudier l'application de trois métaheuristiques - Recuit Simulé, Recherche Tabou et Algorithme Génétique - au problème du sac à dos. D'après tous nos tests, nous avons pu trouver les avantages et inconvénients de ces types de résolutions :

- Le Recuit Simulé a été très rapide et consistant, bien qu'il n'ait pas toujours trouvé la solution optimale, et ce même en changeant ses paramètres.
- Le tabou a pu trouver les solutions optimales dans un seul jeu de données, sinon il se coinçait dans des maximum locaux, et ce peu importe la taille de sa liste de tabou.
- L'algorithme génétique est le plus polyvalent des trois méthodes et est aussi assez rapide. Cependant certains de ces paramètres doivent être initialisés correctement pour donner une bonne solution.