Stéphane COCQUEBERT
William LARCY

Université Polytechnique HAUTS-DE-FRANCE

TP
Métaheuristiques

2020 - 2021

Algorithme

Nous avons choisi d'implémenter l'algorithme génétique

Il prend en entrée le sac vide qui a été créé à partir du fichier du sac, le nombre d'itérations à réaliser, le nombre de sacs que manipuler l'algorithme a chaque itération et un booléen pour savoir si l'on utilise la valeur maximale théorique pour s'arrêter plusieurs tôt, s'il n'y en a pas elle n'est d'office pas pris en compte.

L'algorithme commence par créer une liste avec le nombre de sacs qui sera manipulé durant les itérations et les initialise en prenant pour chaque groupe un objet aléatoire. On initialise la meilleure solution comme étant la solution ou l'on prend aucun objet puis l'algorithme génétique commence.

L'algorithme effectue en premier lieu la **mutation** de chaque sac, la **mutation** que l'on utilise dans notre algorithme est la suivante :

On parcourt la liste symbolisant le choix de l'objet à prendre, si à l'indice i on ne prend aucun objet, nous avons 75% de chance de prendre un objet sinon nous avons 30% de chance que la valeur prenne une nouvelle valeur. La raison qui nous a poussé à augmenter la probabilité de mutation si nous ne prenant aucun objet pour le groupe i est que les tests ont montré que le sac était souvent très largement sous remplis, qu'il restait beaucoup de place pour prendre des objets dans le sac.

Une fois que nous avons effectué la mutation de nos sacs, nous supprimons de la liste les sacs avec une solution non possible, les sacs ou leur capacité actuelle et supérieure à la capacité maximum du sac.

Ensuite nous regardons le meilleur sac pour voir s'il est meilleur que notre meilleur sac actuel, si oui il le remplace.

Ensuite nous **sélectionnons** les 2 meilleurs sacs pour réaliser le **cross-over**, lors du **cross-over** nous prenons la première moitié du génome du sac numéro 1 puis la seconde moitié du sac numéro 2 pour réaliser l'enfant 1 et nous prenons la première moitié du sac numéro 2 et la seconde moitié du sac numéro 1 pour réaliser le sac enfant numéro 2

Pour finir, nous créons la prochaine génération qui sera composée du meilleur sac actuel, des 2 enfants résultant des 2 meilleurs sacs de la génération courante et de x sacs générés aléatoirement (x dépendant du nombre de sacs que l'on souhaite manipuler à chaque génération).

L’algorithme se termine lorsque i itérations on était réalisé, il peut également se terminer plusieurs tôt si nous avons mis en paramètre que l’ont souhaité utiliser la valeur maximale théorique et si celle-ci existe.

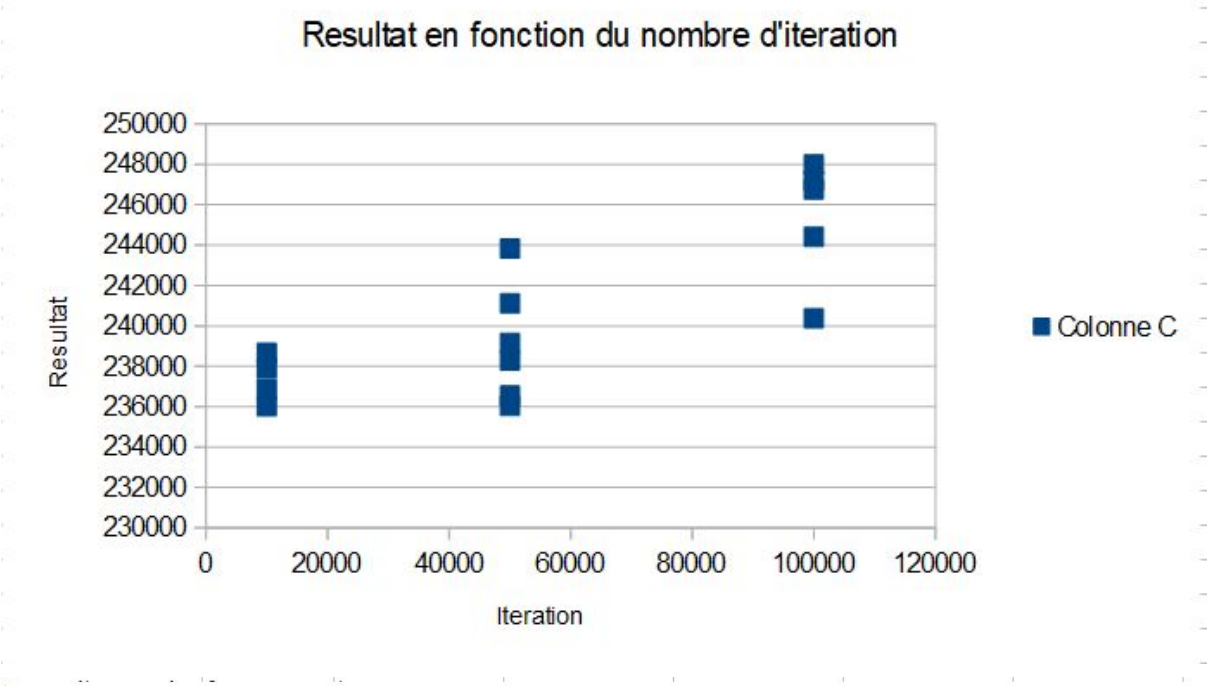
Résultats

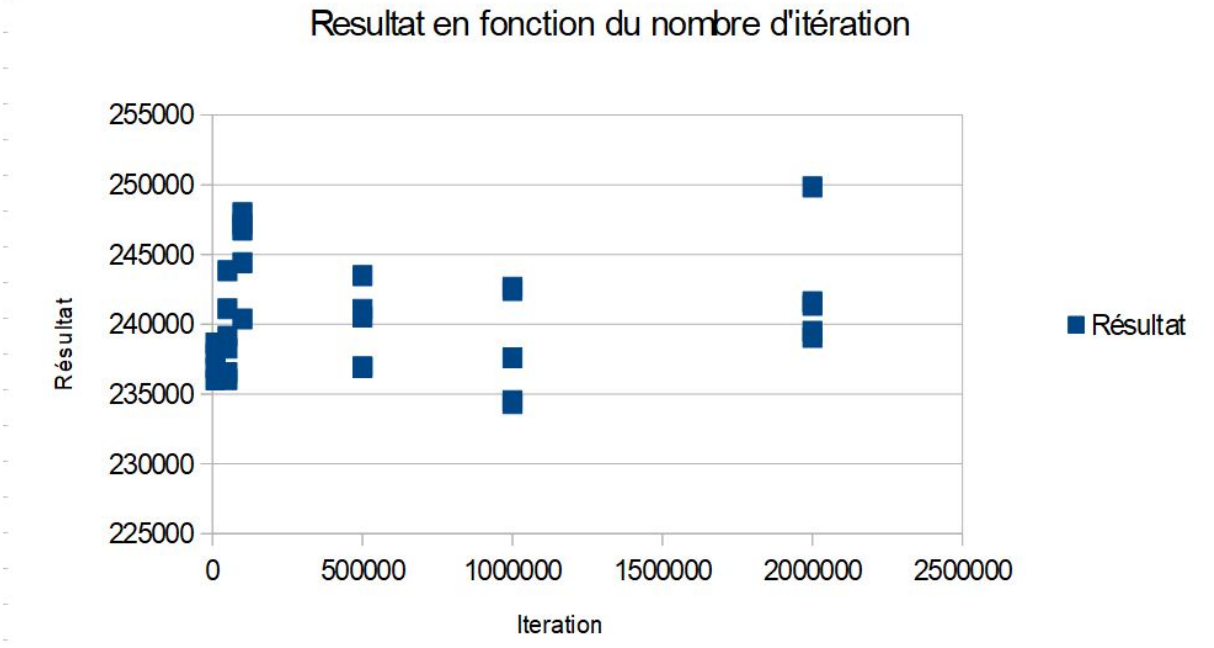
Pour chaque test nous avons manipulé 8 sacs. Le nombre d’itérations est indiqué dans le tableau pour chaque test.

Le tableau ci dessous résume les résultat obtenue sur 30 tests effectué sur le fichier sdkp2_4.txt et ou la valeurs théorique maximum est 340 027:

	Nombre d’itération	Résultat	Capacité utilisé	Capacité restante	temps
1	10000	238657	183961	30683	3s
2	10000	237895	183062	31582	3s
3	10000	236870	179747	34897	3s
4	10000	236024	179752	34892	3s
5	50000	243838	187179	27465	14s
6	50000	236543	179922	34722	14s
7	50000	236046	178614	36030	14s
8	50000	238285	183708	30936	14s
9	50000	239155	180979	33665	14s
10	50000	241118	184890	29754	14s
11	100000	244416	186693	27951	28s
12	100000	246754	187011	27633	28s
13	100000	247193	185402	29242	28s
14	100000	247989	189361	25283	28s
15	100000	240385	180332	34312	28s
16	500000	241075	181952	32692	2m24s
17	500000	236967	181175	33469	2m24s
18	500000	243485	184684	29960	2m24s

19	500000	240521	186307	28337	2m24s
20	500000	236889	181142	33502	2m24s
21	1000000	237606	180099	34545	4m46s
22	1000000	234303	181474	33170	4m43s
23	1000000	242651	186641	28003	4m49s
24	1000000	234532	181541	33103	4m41s
25	1000000	242404	189339	25305	4m37
26	2000000	239048	183166	31478	9m32s
27	2000000	241351	185021	29623	9m32s
28	**2000000**	**249843**	**189219**	**25425**	**9m32s**
29	2000000	239506	181424	33220	9m32s
30	2000000	241613	185575	29069	9m32s





On peut constater que les résultats obtenus avec 100 000 itérations sont plus élevés en moyenne mais que c'est la série résultante effectuée avec 2 000 000 d'itérations qui obtient la meilleure valeur, le test 28 avec un résultat de **249 843.**

Fichier de sauvegarde

Un fichier représentant la solution est généré à la suite de l'exécution de l'algorithme, voici comment il est composé :

- nombre de groupes
- capacité du sac
- Valeur de la solution
- Les coûts des objets (séparé par un espace)
- Les poids des objets (séparé par un espace)
- Le solution normalisé, chaque objet y est représenté, 1 si on le prend 0 sinon
- La solution brute, une valeur représente l'objet que l'on prend pour chaque groupe (-1 aucun objet de pris, 0 : objet 1 est pris, 1 : objet 2 est pris et 2 objet 3 est pris)

Les fichiers sont stockés dans le dossier "Solutions" et portes le nom du fichier utilisé pour leurs créations (exemple : si l'on utilise l'instance "sdkp2_4.txt" le fichier solution s'appellera également sdkp2_4.txt".

/!\ Si un fichier se avec le nom est déjà présent dans le répertoire il sera effacé et remplacé par le nouveau.

Conclusion

Nous avons utilisé pour nos tests des sacs ou une valeur théorique était renseignée pour nous permettre de voir l'efficacité de notre algorithme. En moyenne notre algorithme trouve une solution qui est 30% inférieure à la valeur théorique.

Dans les algorithmes génétiques le cœur de l'algorithme qui déterminera son efficacité est sa fonction de mutation. Notre algorithme est encore perfectible, nous devrions dans un premier temps chercher à trouver la meilleure valeur pour faire muter nos cellules voir trouver d'autre critère pour faire muter une cellule.