Livrable de l'exercice d'implémentation 3 : Battle of metaheuristics

Présentation succincte des choix de mise en œuvre de la métaheuristique concurrente à GRASP appliquée au SPP

Présenter l'algorithme mis en oeuvre. Illustrer sur un exemple didactique (poursuivre avec l'exemple pris en EI1). Présenter vos choix de mise en oeuvre.

Algorithme Génétique

Pour cet exercice d'implémentation, j'ai décidé de mettre en place l'algorithme Génétique ; mon algorithme fonctionne avec 4 fonctions principales :

1. **Sélection des parents :** Cette fonction choisit le meilleur individu entre deux individus créés grâce à la fonction greedy_randomized_construction() ; le choix est fait en comparant leur coût total.
2. **Croisement :** Cette fonction permet de créer deux nouveaux individus à partir de deux parents sélectionnés avec la fonction de sélection des parents. Pour réaliser le croisement on attribue aléatoirement des éléments des parents aux enfants en veillant à ce que l'enfant respecte la contrainte de couverture des éléments.
3. **Mutation :** On garde les éléments communs, puis on complète aléatoirement avec des éléments non conflictuels de l'union des parents pour obtenir un enfant faisable.
4. **Sélection des enfants :** On sélectionne les meilleurs enfants parmi ceux créés, le choix est fait en comparant leur utilité totale.

Expérimentation numérique comparative GRASP vs métaheuristique concurrente

Présenter le protocole d'expérimentation (environnement matériel ; budget de calcul ; condition(s) d'arrêt ; réglage des paramètres).

Les algorithmes ont tourné sur la machine suivante :

— **Système d'exploitation :** Windows 11
— **Processeur :** AMD RYZEN 7 5700X 8-Core
— **Mémoire :** 16Go

0.1 Budget de calcul et conditions d'arrêt

Pour réalisé cette expérimentation, j'ai décidé de procéder comme ceci :

— J'ai selectionné 10 instances plus ou moins grandes du SPP
— J'ai choisi en paramètre de base pour l'algorithme génétique :
  — Taille de la population : 200
  — Nombre de génération : 500
  — Taux de mutation : 0.5
— J'ai choisi de répéter chaque expérimentation 3x pour chaque instances (3x 3 itération de population différente, 3x 3 itération de génération différente, 3x 3 itération de taux de mutation différent)
  — 5 itération ou on fais varier la taille de la population (100, 150, 200)
  — 5 itération ou on fais varier le nombre de génération (100, 250, 500)

— 5 itération ou on fais varier le taux de mutation (0.1, 0.5, 0.7)

Rapporter graphiquement vos résultats selon $\hat{z}_{min}$, $\hat{z}_{max}$, $\hat{z}_{moy}$ mesurés à intervalles réguliers (exemple de pas de 10 secondes).

1 Graphique des résultats

Les résultats sont obtenu avec 3 runs de chaque instances avec les paramètres "classique" défini plus haut.

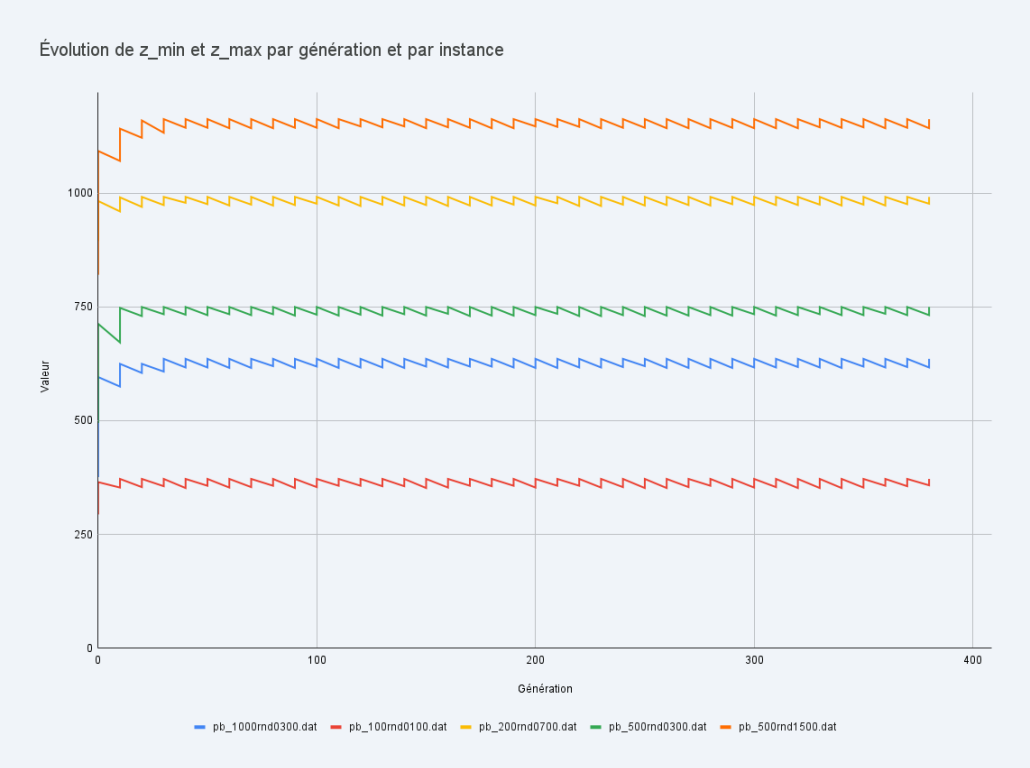

FIGURE 1 – Z_mean.

Rapporter l'étude de l'influence du paramètre α.

Concernant l'Algorithme Génétique (AG), l'influence de ses propres paramètres (étudiés dans le tableau 1) peut être résumée comme suit :

- **Taille de la population :** Augmenter la population (ex : 100 à 200) a l'impact le plus direct sur le **temps de calcul** (parfois doublé, cf. `pb_200rnd0100.dat`), mais son influence sur la qualité de la solution semble plafonner rapidement (rendements décroissants).
- **Nombre de générations :** Un nombre plus élevé permet à l'algorithme de converger. L'analyse montre que l'essentiel de l'amélioration est souvent obtenu avant 500 générations, suggérant une convergence ou stagnation par la suite pour un coût en temps linéaire.
- **Taux de mutation :** Ce paramètre est apparu comme **critique pour la qualité**. Un taux trop faible (ex : 0.1) résulte en des solutions de moins bonne qualité (ex : moyenne de 412 pour `pb_200rnd0100.dat`) comparé à des taux plus élevés (0.5 ou 0.7, moyenne 404), car il ne permet pas d'échapper aux optima locaux.

Présenter sous forme de tableau les résultats finaux obtenus pour les 10 instances sélectionnées.

TABLE 1: Résultats détaillés de l'algorithme génétique par groupe de paramètres.

Instance	Run Group	Run ID	Repeat	Pop.	Gens.	Mut.	Best	Time (s)
dat/pb_100rnd0100.dat	pop_size	1	1	100	500	0.5	372	2.085
			2				372	1.87
			3				372	1.872
		2	1	150	500	0.5	372	2.825
			2				372	2.789
			3				372	3.18
		3	1	200	500	0.5	372	4.425
			2				372	4.427
			3				372	4.472
	generations	4	1	200	100	0.5	372	1.089
			2				372	1.031
			3				372	1.018
		5	1	200	250	0.5	372	2.313
			2				372	2.299
			3				372	2.259
		6	1	200	500	0.5	372	4.36
			2				372	4.383
			3				372	4.394
	mutation	7	1	200	500	0.1	372	3.518
			2				372	3.587
			3				372	3.604
		8	1	200	500	0.5	372	4.35
			2				372	4.335
			3				372	4.296
		9	1	200	500	0.7	372	4.805
			2				372	4.706
			3				372	4.798
dat/pb_200rnd0100.dat	pop_size	1	1	100	500	0.5	409	4.075
			2				408	3.916
			3				408	3.635
		2	1	150	500	0.5	404	5.569
			2				416	6.716
			3				409	6.217
		3	1	200	500	0.5	412	7.481
			2				403	7.677
			3				410	8.065
	generations	4	1	200	100	0.5	400	1.862
			2				414	2.08
			3				403	1.895
		5	1	200	250	0.5	399	4.175
			2				403	3.912
			3				410	4.292
		6	1	200	500	0.5	403	7.618
			2				408	8.045
			3				412	7.476
	mutation	7	1	200	500	0.1	414	5.702
			2				414	5.727
			3				408	6.336
		8	1	200	500	0.5	399	7.634
			2				407	7.588
			3				408	7.179

Table 1: Résultats détaillés de l'algorithme génétique par groupe de paramètres (suite)

Instance	Run Group	Run ID	Repeat	Pop.	Gens.	Mut.	Best	Time (s)
		9	1	200	500	0.7	416	8.948
			2				399	8.201
			3				399	8.533
dat/pb_500rnd0300.dat	pop_size	1	1	100	500	0.5	744	16.545
			2				754	16.537
			3				730	15.946
		2	1	150	500	0.5	737	23.973
			2				740	24.598
			3				754	24.311
		3	1	200	500	0.5	744	32.789
			2				750	32.248
			3				754	33.185
	generations	4	1	200	100	0.5	750	9.565
			2				748	10.161
			3				750	10.005
		5	1	200	250	0.5	744	18.715
			2				740	18.231
			3				755	18.412
		6	1	200	500	0.5	752	32.737
			2				750	32.339
			3				754	33.822
	mutation	7	1	200	500	0.1	752	25.956
			2				751	26.721
			3				740	27.121
		8	1	200	500	0.5	754	36.276
			2				742	34.981
			3				754	35.492
		9	1	200	500	0.7	763	39.277
			2				732	39.005
			3				754	39.036
dat/pb_500rnd1500.dat	pop_size	1	1	100	500	0.5	1155	16.316
			2				1144	16.623
			3				1151	16.741
		2	1	150	500	0.5	1160	25.857
			2				1131	25.39
			3				1165	26.802
		3	1	200	500	0.5	1169	39.94
			2				1140	35.552
			3				1148	33.423
	generations	4	1	200	100	0.5	1162	10.513
			2				1162	10.668
			3				1165	10.434
		5	1	200	250	0.5	1152	19.124
			2				1146	19.045
			3				1151	18.949
		6	1	200	500	0.5	1137	33.501
			2				1151	33.343
			3				1162	32.292
	mutation	7	1	200	500	0.1	1164	24.817
			2				1141	24.396
			3				1148	25.201

TABLE 1: Résultats détaillés de l'algorithme génétique par groupe de paramètres (suite)

Instance	Run Group	Run ID	Repeat	Pop.	Gens.	Mut.	Best	Time (s)
		8	1	200	500	0.5	1162	31.844
			2				1155	31.26
			3				1162	31.533
		9	1	200	500	0.7	1172	34.537
			2				1162	35.406
			3				1160	38.371
dat/pb_500rnd1700.dat	pop_size	1	1	100	500	0.5	168	2.572
			2				168	2.292
			3				168	2.509
		2	1	150	500	0.5	185	3.74
			2				175	3.467
			3				173	3.698
		3	1	200	500	0.5	185	5.241
			2				185	5.15
			3				192	5.063
	generations	4	1	200	100	0.5	184	1.394
			2				179	1.669
			3				163	1.354
		5	1	200	250	0.5	174	2.829
			2				188	2.931
			3				170	3.302
		6	1	200	500	0.5	176	4.761
			2				167	4.565
			3				182	5.183
	mutation	7	1	200	500	0.1	174	3.881
			2				187	4.388
			3				171	3.828
		8	1	200	500	0.5	188	5.251
			2				185	5.318
			3				187	5.568
		9	1	200	500	0.7	176	5.205
			2				182	5.651
			3				185	6.057
dat/pb_200rnd0400.dat	pop_size	1	1	100	500	0.5	62	8.658
			2				62	8.951
			3				62	8.792
		2	1	150	500	0.5	62	13.227
			2				62	13.149
			3				62	12.844
		3	1	200	500	0.5	63	17.865
			2				62	17.898
			3				62	17.552
	generations	4	1	200	100	0.5	62	4.525
			2				62	4.367
			3				62	4.594
		5	1	200	250	0.5	62	9.399
			2				62	9.809
			3				62	9.571
		6	1	200	500	0.5	62	17.818
			2				62	18.936
			3				62	18.904

Table 1: Résultats détaillés de l'algorithme génétique par groupe de paramètres (suite)

Instance	Run Group	Run ID	Repeat	Pop.	Gens.	Mut.	Best	Time (s)
			1				62	14.524
		7	2	200	500	0.1	62	14.912
			3				62	15.199
			1				62	18.753
	mutation	8	2	200	500	0.5	62	19.039
			3				62	19.061
			1				62	21.029
		9	2	200	500	0.7	62	19.966
			3				62	20.336
			1				990	3.417
		1	2	100	500	0.5	997	3.451
			3				991	3.512
			1				991	5.452
	pop_size	2	2	150	500	0.5	989	5.152
			3				991	5.096
			1				992	6.857
		3	2	200	500	0.5	992	6.802
			3				992	7.186
			1				993	2.027
		4	2	200	100	0.5	994	1.604
dat/pb_200rnd0700.dat			3				991	1.61
			1				995	3.446
	generations	5	2	200	250	0.5	989	3.386
			3				989	3.441
			1				992	6.527
		6	2	200	500	0.5	986	6.435
			3				992	6.287
			1				991	5.655
		7	2	200	500	0.1	991	5.573
			3				991	5.682
			1				989	6.831
	mutation	8	2	200	500	0.5	991	6.603
			3				989	6.166
			1				989	6.514
		9	2	200	500	0.7	992	6.998
			3				990	7.028
			1				55	3.239
		1	2	100	500	0.5	67	3.577
			3				67	3.578
			1				67	5.155
	pop_size	2	2	150	500	0.5	59	4.366
			3				67	5.143
			1				57	5.661
		3	2	200	500	0.5	67	6.674
			3				67	6.724
			1				59	2.298
		4	2	200	100	0.5	59	2.533
dat/pb_1000rnd0100.dat			3				57	2.385
			1				56	3.557
	generations	5	2	200	250	0.5	67	4.089
			3				59	3.578

TABLE 1: Résultats détaillés de l'algorithme génétique par groupe de paramètres (suite)

Instance	Run Group	Run ID	Repeat	Pop.	Gens.	Mut.	Best	Time (s)
			1				67	6.717
		6	2	200	500	0.5	67	6.715
			3				67	6.804
			1				57	4.606
		7	2	200	500	0.1	67	5.417
			3				59	4.444
	mutation		1				67	6.782
		8	2	200	500	0.5	67	6.668
			3				67	6.661
			1				59	6.144
		9	2	200	500	0.7	57	6.345
			3				59	6.175
			1				589	24.005
		1	2	100	500	0.5	581	24.138
			3				593	24.226
			1				599	34.225
	pop_size	2	2	150	500	0.5	604	35.242
			3				593	31.314
			1				579	41.95
		3	2	200	500	0.5	592	42.788
			3				588	42.55
			1				598	16.965
		4	2	200	100	0.5	608	16.777
dat/pb_1000rnd0300.dat			3				596	16.58
	generations		1				616	27.418
		5	2	200	250	0.5	625	27.736
			3				595	26.014
			1				597	48.604
		6	2	200	500	0.5	592	49.824
			3				608	52.782
			1				633	41.84
		7	2	200	500	0.1	604	38.161
			3				612	40.256
	mutation		1				609	45.487
		8	2	200	500	0.5	617	42.64
			3				590	50.089
			1				599	54.801
		9	2	200	500	0.7	594	56.423
			3				598	53.568
			1				30	0.149
		1	2	100	500	0.5	30	0.122
			3				30	0.132
			1				30	0.2
	pop_size	2	2	150	500	0.5	30	0.178
			3				30	0.182
			1				30	0.251
		3	2	200	500	0.5	30	0.242
			3				30	0.269
			1				30	0.068
		4	2	200	100	0.5	30	0.07
dat/didactic.dat			3				30	0.059
	generations							

TABLE 1: Résultats détaillés de l'algorithme génétique par groupe de paramètres (suite)

Instance	Run Group	Run ID	Repeat	Pop.	Gens.	Mut.	Best	Time (s)
		5	1	200	250	0.5	30	0.137
			2				30	0.13
			3				30	0.127
		6	1	200	500	0.5	30	0.236
			2				30	0.264
			3				30	0.244
	mutation	7	1	200	500	0.1	30	0.24
			2				30	0.245
			3				30	0.244
		8	1	200	500	0.5	30	0.257
			2				30	0.276
			3				30	0.249
		9	1	200	500	0.7	30	0.258
			2				30	0.269
			3				30	0.26

Discussion

Tirer des conclusions en comparant les résultats collectés avec vos deux métaheuristiques.
En comparant les résultats obtenus avec GRASP et l'algorithme génétique, on observe un point commun : les deux méthodes tendent à produire de meilleures solutions lorsque que l'aléatoire est important (alpha élevé pour GRASP, taux de mutation élevé pour l'algorithme génétique). Toutefois, l'algorithme génétique présente, dans notre étude, de meilleures performances sur la majorité des instances testées. Cet avantage est particulièrement visible pour les instances de grande taille (par exemple `pb_1000rnd0300.dat`).
Quelles sont les recommandations que vous émettez à l'issue de l'étude ?
Sur la base des expériences menées, je recommande l'utilisation de l'algorithme génétique : il a fourni les meilleurs résultats dans la plupart des cas étudiés. Il reste néanmoins possible d'améliorer encore ses performances en affinant la sélection des parents et les opérateurs de croisement. Il faut aussi garder à l'esprit un inconvénient pratique : l'algorithme génétique comporte plusieurs hyperparamètres (taille de la population, nombre de générations, taux de mutation), ce qui rend le réglage plus exigeant que pour GRASP, dont le paramètre principal est α (ou la liste d'alpha en Reactive GRASP). En conséquence, si le temps et les ressources de calibration sont limités, GRASP reste une option intéressante pour sa simplicité.