Algorithmes génétiques et problème du sac-à-dos avec plusieurs contraintes de coût.

Yang YANG & Binh-Minh NGUYEN

Université Paris Dauphine

M1 MIAGE en apprentissage

Sommaire

I. Choix des paramètres expérimentaux et configuration de l'ordinateur

Nous pouvons définir plusieurs paramètres importants pour le problème du sac à dos à l'aide de la classe ParameterSetting. Cela permet de définir différents problèmes de sac à dos en modifiant **le nombre d'objets, le nombre de contraintes de coût** et **la taille de la population**. Les valeurs du budget ainsi que les coûts et utilités de chaque objet sont générées aléatoirement par l'algorithme.
Dans la méthode run_multiple_instances de la classe GeneticAlgorithm, nous utilisons les paramètres suivants :

- **nombre_iteration** : Détermine à quelle génération l'expérience s'arrête.
- **nombre_instance** : Définit le nombre de fois que chaque algorithme génétique est exécuté pour éviter les résultats aléatoires.

A) Définition du problème

Pour le test, nous avons choisi les paramètres suivants :

- Nombre d'objets (nombre d'objet) : 10
- Nombre de contraintes de coût (nombre de contraintes de coût) : 3
- Taille de la population (population size) : 20

Voici le scénario :
Liste de budgets :
[137 112 172]
Liste d'objets :

	utilities	cost_1	cost_2	cost_3
0	10	19	6	19
1	76	12	11	15
2	6	19	5	10
3	80	18	1	14
4	65	10	10	8
5	17	2	1	18
6	2	9	14	16
7	77	11	9	8
8	72	4	7	18
9	7	4	5	18

B) Nombre d'itération et instance

Pour les tests, nous avons choisi :

- nombre_iteration : 100
- nombre_instance : 10

C) Environnement d'exécution

Les données de cette expérience ont été obtenues dans l'environnement suivant :
Nom : MacBook Pro (16 pouces, 2019)
Processeur : 2,3 GHz Octa-core Intel Core i9
Carte graphique : Intel UHD Graphics 630 avec 1536 MB de VRAM
Mémoire : 16 GB de RAM DDR4 à 2667 MHz
Système d'exploitation : macOS Sonoma version 14.4.1
Stockage : 1 TB SSD

II. Variables de l'expérience - 8 stratégies (Variantes des algorithmes de sélection, croisement et mutation)

Nous gardons l'algorithme génétique global inchangé, c'est-à-dire que l'initialisation de la population, la fonction de réparation et la logique de l'algorithme génétique restent les mêmes. En combinant deux méthodes de sélection, deux méthodes de croisement et deux méthodes de mutation, nous obtenons 2 * 2 * 2 = 8 stratégies.

- Méthodes de sélection
  1 : select_fitness_proportional
  Chaque individu a une probabilité d'être sélectionné comme parent proportionnelle à son fitness. Plus le fitness est élevé, plus la probabilité est grande.
  2 : select_tournament
  Sélectionne aléatoirement un sous-groupe d'individus (ici, taille du sous-groupe = 5) et choisit l'individu avec le fitness la plus élevée.

- Méthodes de croisement
  1 : crossover_random
  Chaque gène a 50% de chance de provenir du parent 1 et 50% de chance de provenir du parent 2.
  2 : crossover_two_point
  Sélectionne deux points aléatoires, coupe les deux parents en trois segments et combine les segments pour créer un nouvel individu.

- Méthodes de mutation
  1 : mutate_single_point
  Sélectionne aléatoirement un gène et inverse sa valeur (0 devient 1 et 1 devient 0).
  2 : mutate_fitness_proportional
  Calcule la fitness moyenne de la population. Pour chaque individu :
  Si le fitness est supérieur à la moyenne : taux de mutation de chaque gène = 0.1
  Si le fitness est inférieur à la moyenne : taux de mutation de chaque gène = 0.2

Les huit stratégies résultantes :

strategy1:
select_fitness_proportional, crossover_random, mutate_single_point
strategy2:
select_fitness_proportional, crossover_random, mutate_fitness_proportional
strategy3:
select_fitness_proportional, crossover_two_point, mutate_single_point
strategy4:
select_fitness_proportional, crossover_two_point, mutate_fitness_proportional
strategy5:
select_tournament, crossover_random, mutate_single_point
strategy6:
select_tournament, crossover_random, mutate_fitness_proportional
strategy7:
select_tournament, crossover_two_point, mutate_single_point
strategy8:
select_tournament, crossover_two_point, mutate_fitness_proportional

III. Critères d'évaluation des stratégies

Pour nos huit stratégies, nous comparons à partir des quatre paramètres suivants :

1. Temps d'exécution moyen :
    - Mesurer le temps moyen nécessaire pour exécuter chaque stratégie.
2. Fitness maximal moyenne après 100 itérations :
    - Calculer le fitness maximal moyenne obtenue après 100 itérations.
3. Tendance de la variation du fitness maximale moyenne en fonction des itérations :
    - Observer et comparer la tendance de la variation du fitness maximale moyenne au fil des itérations pour chaque stratégie.
4. Écart-type du fitness maximal à chaque itération sur 10 exécutions :
    - Calculer l'écart-type du fitness maximal à chaque itération pour 10 exécutions de chaque stratégie.

Pour les points 1 et 2, nous pouvons directement afficher les résultats. Pour les points 3 et 4, nous pouvons tirer des conclusions à partir des graphiques générés.

IV. Comparaison des résultats expérimentaux (Présentation des graphiques et des conclusions)

A) Fitness maximale moyenne et temps d'exécution

```
Testing strategy1: select_fitness_proportional, crossover_random, mutate_single_point
Max Average Fitness: 405.0
Elapsed Time: 6.28 seconds
----------------------------------------------------------------
Testing strategy2: select_fitness_proportional, crossover_random,
mutate_fitness_proportional
Max Average Fitness: 406.9
Elapsed Time: 11.26 seconds
----------------------------------------------------------------
Testing strategy3: select_fitness_proportional, crossover_two_point, mutate_single_point
Max Average Fitness: 404.7
Elapsed Time: 6.36 seconds
----------------------------------------------------------------
Testing strategy4: select_fitness_proportional, crossover_two_point,
mutate_fitness_proportional
Max Average Fitness: 407.2
Elapsed Time: 11.06 seconds
----------------------------------------------------------------
Testing strategy5: select_tournament, crossover_random, mutate_single_point
Max Average Fitness: 411.8
Elapsed Time: 2.69 seconds
----------------------------------------------------------------
Testing strategy6: select_tournament, crossover_random, mutate_fitness_proportional
Max Average Fitness: 412.0
Elapsed Time: 7.37 seconds
----------------------------------------------------------------
Testing strategy7: select_tournament, crossover_two_point, mutate_single_point
Max Average Fitness: 412.0
Elapsed Time: 3.12 seconds
----------------------------------------------------------------
Testing strategy8: select_tournament, crossover_two_point, mutate_fitness_proportional
Max Average Fitness: 412.0
Elapsed Time: 7.91 seconds
----------------------------------------------------------------
```

Fitness maximal moyenne

Nous pouvons observer que les stratégies 5, 6, 7 et 8 présentent un fitness maximal moyenne significativement supérieure à celles des stratégies 1, 2, 3 et 4. En particulier, les stratégies 6, 7 et 8 atteignent toutes une valeur maximale de 412, indiquant que la méthode de sélection par tournoi (**select_tournament**) est nettement plus efficace pour obtenir un fitness maximal que la méthode proportionnelle à la fitness (**select_fitness_proportional**).

Temps de calcul

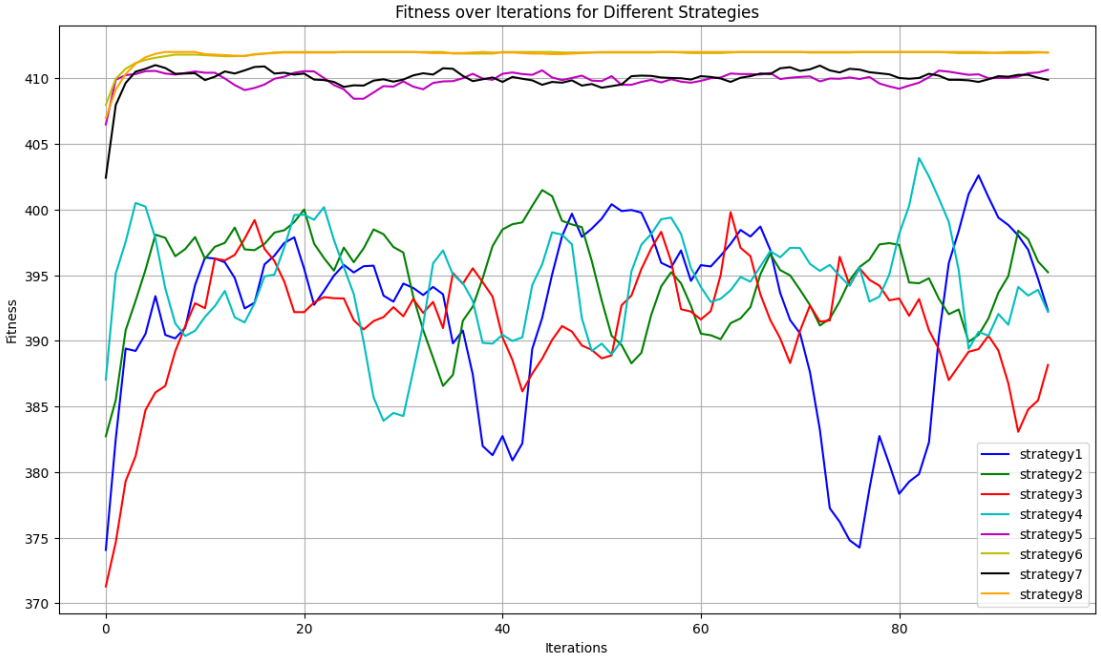
1. **Temps de calcul les plus courts** :
    - **Strategy5** : 2.69 secondes
    - **Strategy7** : 3.12 secondes

2. **Temps de calcul moyens** :
    - **Strategy3** : 6.36 secondes
    - **Strategy6** : 7.37 secondes
    - **Strategy8** : 7.91 secondes
3. **Temps de calcul les plus longs** :
    - **Strategy4** : 11.06 secondes
    - **Strategy2** : 11.26 secondes

Nous pouvons noter que pour des stratégies utilisant les mêmes méthodes de sélection et de croisement, celles utilisant **mutate_fitness_proportional** nécessitent un temps de calcul significativement plus long que celles utilisant **mutate_single_point**.

B) Tendance de la variation du fitness maximale moyenne en fonction des itérations

Pour commencer, nous comparons la tendance globale des huit stratégies. Étant donné que l'ajout des écarts-types dans le graphique global dégrade l'effet visuel, l'axe vertical ici ne représente que le fitness maximal moyenne. Les écarts-types seront illustrés dans des graphiques individuels pour chaque stratégie, que vous pourrez voir dans la prochaine section intitulée "Comparaison des écarts-types".



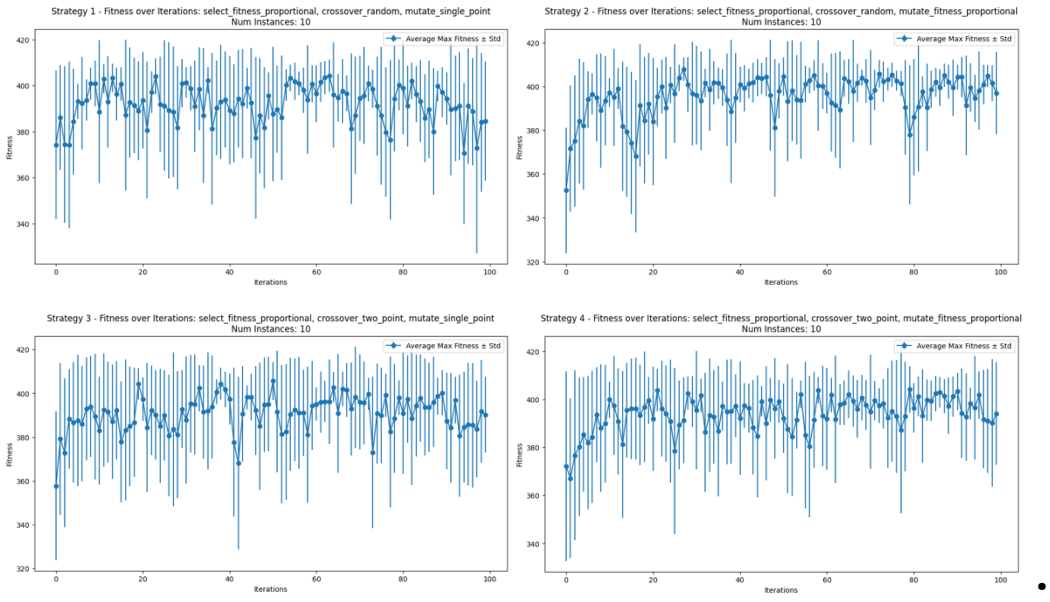
Les graphiques montrent clairement que **les stratégies 5, 6, 7 et 8** présentent une tendance significative à **l'augmentation** du fitness maximale moyenne calculée sur 10 instances au fur et à mesure des itérations. En revanche, pour **les stratégies 1, 2, 3 et 4**, le fitness maximal moyenne calculée sur 10 instances **fluctue** avec l'augmentation des itérations. Cela confirme encore une fois la conclusion précédente selon laquelle la méthode de sélection par tournoi **(select_tournament**) est nettement supérieure à la méthode proportionnelle à la fitness **(select_fitness_proportional)**

pour obtenir un fitness maximal.

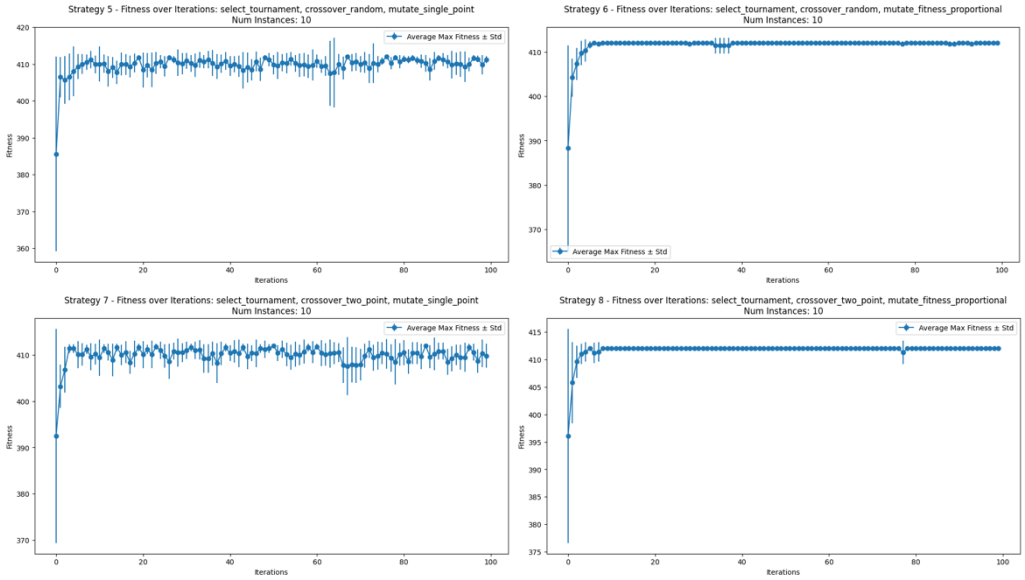
Ensuite, nous comparons **les quatre courbes des stratégies 5, 6, 7 et 8. Les stratégies 6 et 8** se distinguent comme étant les plus performantes et les plus stables. Elles atteignent très rapidement la fitness maximale (412) et maintiennent cette valeur au cours des itérations suivantes. Les deux utilisent la combinaison **select_tournament** et **mutate_fitness_proportional,** ce qui nous permet de conclure que la méthode mutate_fitness_proportional est plus efficace que mutate_single_point pour générer des populations plus performantes.

C) Comparaison des écarts-types

Nous avons généré une image pour chaque stratégie, avec l'axe horizontal représentant le nombre d'itérations et l'axe vertical représentant **le fitness maximal moyenne ± l'écart-type** sur 10 exécutions. Comme il y a huit images, nous avons inséré des images plus petites dans ce rapport. Vous pouvez trouver les images haute résolution dans le fichier “image”.



Nous pouvons observer que pour **les quatre premières stratégies**, avec l'augmentation des itérations, non seulement **le fitness maximal moyenne fluctue**, mais aussi **l'écart-type du fitness maximal est relativement grand**. Cela indique une forte instabilité des algorithmes de ces quatre stratégies.



Pour **les quatre dernières stratégies**, avec l'augmentation des itérations, **le fitness maximal moyenne montre une tendance à la hausse**. Les stratégies 5 et 7 présentent **une relative grande variabilité dans l'écart-type du fitness maximal** à chaque itération, indiquant une certaine instabilité. En revanche, **les stratégies 6 et 8** deviennent **stables** après avoir atteint le fitness maximal, avec **un écart-type très faible**. Parmi elles, la stratégie 8 se distingue par sa stabilité la plus forte.

V. Conclusion

Après avoir analysé les performances des différentes stratégies, nous avons identifié **les stratégies 6 (select_tournament, crossover_random, mutate_fitness_proportional)** et **8 (select_tournament, crossover_two_point, mutate_fitness_proportional)** comme étant les plus fortes.

Stratégie 6 présente **un léger avantage en termes de temps de calcul** avec une moyenne de 7.37 secondes par exécution, ce qui en fait une option légèrement plus rapide. Cependant, **Stratégie 8** montre **une plus grande stabilité dans les résultats, avec un écart-type plus faible** après avoir atteint le fitness maximal. Cela signifie que les résultats obtenus avec la stratégie 8 sont un peu plus fiables.

Ces deux stratégies utilisent la sélection par tournoi et la mutation proportionnelle au fitness, soulignant l'efficacité de ces méthodes pour obtenir des solutions optimales et stables dans les problèmes d'optimisation avec contraintes multiples.