TP 2

Résolution du Problème FloatPairs avec JMetalPy

Dans ce TP, vous allez implémenter le problème FloatPairs. Pour vous faciliter l'implémentation, nous vous proposons le problème FloatSum pour vous inspirer.

Introduction : FloatSum (pour s'inspirer) Le problème de FloatSum est un problème d'optimisation jouet cherchant à maximiser la somme des **n** nombres flottants composant la solution. Ces nombres flottants sont compris entre [borne_inf , borne_sup]

Par exemple, pour n=7, bi = -273.15 et bs = 5500.00

la solution [10.5, -26.3, 29.35, 12.22, -13.51, 2.24, 52.07] a un score de 66,57
la solution [5476.33, -227.06, 2896.86, 5212.22, 3913.51, 4302.24, 3352.07] a un score de 24926,17

Dans cet exemple, la deuxième solution est de meilleure qualité que la première solution. La meilleure solution serait [5500.00, 5500.00, 5500.00, 5500.00, 5500.00, 5500.00, 5500.00]

Introduction : FloatPairs (à implémenter) Le problème FloatPairs est un problème d'optimisation jouet cherchant à maximiser le comptage des paires de flottants de signes différents dans une séquence de flottant donnée. Une paire de flottant est définie comme deux flottants de signes différents, par exemple -5.0 et 3.0, ou 6.0 et -2.5

Comme pour FloatSum, une solution comporte **n** nombres flottants compris entre [borne_inf , borne_sup].

Voici maintenant un exemple de solution pour le FloatPairs pour n=8, bi = -10.00 et bs = 10.00

La solution [-10.00, 5.00, 6.25, 3.14, -8.7, -9.75, 1.36,-9.99] comporte les paires suivantes :

- -10.00 et 5.00 (positions 1 et 2)
- 3.14 et -8.7 (positions 4 et 5)
- -9.75 et 1.36 (positions 6 et 7)
- 1.36 et -9.99 (positions 7 et 8)

Il y a donc un total de 4 paires de signes différent adjacentes dans la séquence donnée. Le score sera donc de 4.

Rendu : Vous rendrez un rapport sur l'ensemble des questions de ce TP et les codes python correspondant dans un fichier zip. Le rendu peut se faire par binôme.

Le Problème FloatPairs dans JMetalPy :

Le problème FloatPairs n'est pas implémenté dans JMetalPy, nous allons devoir le créer.
Pour cela regardez https://jmetal.github.io/jMetalPy/tutorials/problem.html

Voici le problème défini pour le **FloatSum**

```python
class FloatSumMax(Problem):
    def __init__(self, number_of_floats, min_value, max_value):
        super().__init__()

        self.number_of_floats = number_of_floats
        self.min_value = min_value
        self.max_value = max_value

        self.lower_bound = [self.min_value] * number_of_floats
        self.upper_bound = [self.max_value] * number_of_floats

        self.number_of_objectives = 1

        self.obj_directions = [self.MINIMIZE]
        self.obj_labels = ['SumMax']

    def number_of_variables(self) -> int:
        return self.number_of_floats

    def number_of_objectives(self) -> int:
        return 1

    def number_of_constraints(self) -> int:
        return 0


    def evaluate(self, solution) :
        solution.objectives[0] = -sum(solution.variables)
        return solution

    def create_solution(self):
        new_solution = FloatSolution(
            self.lower_bound, self.upper_bound,
self.number_of_objectives, self.number_of_constraints()
        )
        new_solution.variables = [
            random.uniform(self.lower_bound[i] * 1.0,
self.upper_bound[i] * 1.0)
            for i in range(self.number_of_variables())
        ]
        return new_solution

    def name(self):

        return 'FloatSumMax'
```

Une recherche Local pour le **FloatPairs**:

- Adaptez votre code pour créer une recherche locale pour le FloatPairs. Note : le type de solution a changé par rapport au OneMax : il ne s'agit plus d'une chaîne binaire mais d'une liste de nombres flottants, il faut donc changer le type de mutation. Utilisez PolynomialMutation
- Expérimentez les différents paramètres de la recherche locale.
- Faites 20 runs de cette recherche locale, présentez une synthèse des résultats obtenus (moyenne, médiane, écart type, temps de calcul de chaque run)

Un Algorithme Génétique Pour le FloatPairs:

- Adaptez votre code pour créer un algorithme génétique pour le FloatPairs. Note : le type de solution a changé par rapport au OneMax : il ne s'agit plus d'une chaîne binaire mais d'une liste de nombres flottants, il faut donc changer le type de mutation et de croisement. Utilisez PolynomialMutation et SBXCrossOver
- Expérimentez les différents paramètres pour l'algorithme génétique.
- Faitez 20 runs de l'algorithme génétique pour différents paramètres, présentez une synthèse des résultats obtenus (moyenne, médiane, écart type, temps de calcul de chaque run). Qu'observez-vous ?

Analyse des Résultats :

- Collectez et analysez les résultats de vos expériences. Quelle est l'influence des différents paramètres sur la convergence de l'algorithme ?
- Comment comparer équitablement les algorithmes ? Comparez l'algorithme génétique et la recherche locale.
- Proposez un protocole expérimental et réalisez-le.

Rapport :
Le rapport présentera l'ensemble des résultats obtenus et des expérimentations réalisées.