FOURNY Nicolas

BLAMPAIN Maxime

RAPPORT DE PROJET

Résolution de problèmes d'optimisation sur le MixedQAP :

2020-2021

Première partie

Optimisation combinatoire

Question 1.a : Algorithme Hill-Climbing Best-Improvement

L'algorithme Hill-Climbing ou méthode d'escalade est une méthode d'optimisation permettant de trouver un optimum local parmi un ensemble de configurations.

Ci-dessous l'algorithme du Hill-Climbing Best-Improvement

```
Début
Choisir une solution initiale x E X
Evaluer x avec f
optimumLocal <- False

while time < timeLimit && optimumLocal is false
        Choisir x' E V(x) telle que f(x') est max
        If x' strictement meilleur que x
        X <- x'
        Else
        optimumLocal <- true
        end if
end
return x
Fin
```

La sélection de voisinage est défini par l'échange de 2 élèments (swap) comme demandé dans la question.

Question 1.b : Algorithme Iterated Local Search

L'algorithme Iterated Local Search (ILS) est une méthode générale utilisée pour résoudre des problèmes d'optimisation, c'est-à-dire des problèmes où l'on cherche la meilleure solution dans un ensemble de solutions candidates.
La recherche locale consiste à passer d'une solution à une autre solution proche dans l'espace des solutions candidates (l'espace de recherche) jusqu'à

ce qu'une solution considérée comme optimale soit trouvée, ou que le temps imparti soit dépassé.

Debut
Choisir une solution initiale x E X
X <- HillClimbing(x)
While time < timeLimit
X’ <- perturbation(x)
X’ <- HillClimbing(x’)

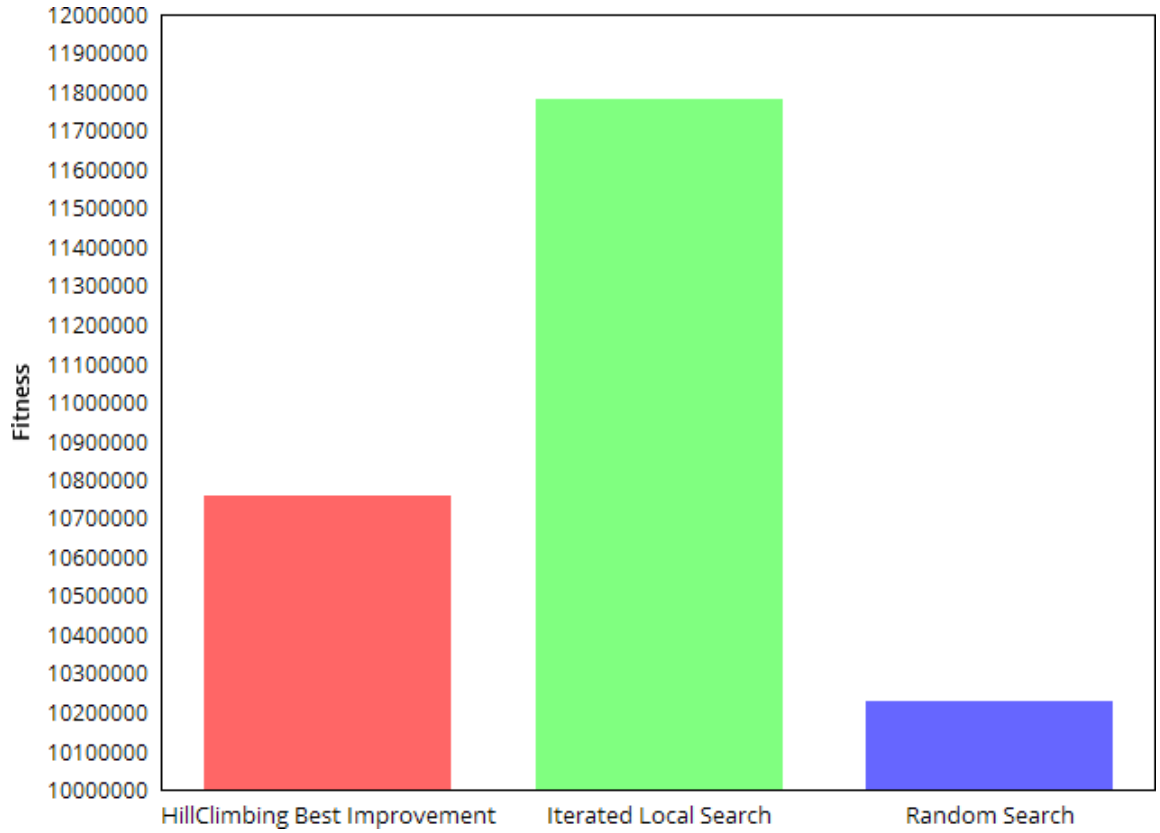
If x’ strictement meilleur que x
X <- X’
End if
End
Fin

Question 1.d : évaluer les performances de votre algorithme sur les instances fournies

Nous avons répété pour chaque instance 30 fois chaque algorithmes pour obtenir une donnée que l’on peut comparer comme nous l’avions entendu en cours (minimum requis).

Nous avons créés des scripts BASH pour simplifier le stockage des données dans un csv pour chaque algorithme. Il est indiqué dans chaque csv : l’instance, le seed et la fitness.

Pour représenter les performances nous avons utilisé un script python pour afficher les données sous forme de plot.



Nous pouvons facilement constater que le HillClimbing et Iterated Local Search sont plus efficace qu'une recherche aléatoire.
La moyenne est sensiblement identique mise à part pour la recherche aléatoire qui est moins performant.

Deuxième partie

Optimisation continue

Question 1.a : Algorithme EsSimple (1 + 1)-ES

Les stratégies d'évolution (ES) sont un type d'algorithme d'optimisation adaptées pour les problèmes boîte noire. Ce sont des algorithmes robustes, à envisager lorsqu'il n'y a pas de gradient disponible ou que celui-ci serait difficile d'usage du fait de problèmes particuliers (présence de bruit, de plateaux ou de points de selles).

Voici le pseudo code inspiré du cours de M.Chotard.

```
Debut
sigma=0.0005
gamma=1.1
itérations = 0
While time < timeLimit && iterations < 200
        cumul = 0
        Choisir une solution initiale x E X
        For i < solution.size()
                Random = distribution(_rng) ;

                solution.x[i] += sigma * random
                if (solution.x[i] < 0)
                        sol.x[i] *= -1

                Cumul += sol.x[i] ;

        If cumul == 0
                Cumul = 0
                For i < solution.size()
                        Solution.x[i] = 1/ sol.x.size()
                        Cumul += sol.x[i]
        Else if cumul != 1
                For i < solution.size()
                        Solution.x[i] / cumul

        Sol.modifiedX = true
        Eval(solution)
```

```
If (sol.fitness < solution.fitness)
        Solution = sol
        Iterations = 0
        Sigma = sigma * gamma ;
Else
        Iterations++ ;
        Sigma = sigma * pow (gamma, -1/4)
```

Pour représenter les performances nous avons utilisé un script python pourafficher les données sous forme de plot.