

Quadratic Assignement Problem

rapport d’optimisation discrète

ganne | Optimisation discrète | 2019

# Instances de Taillard

Les instances de Taillard qui nous sont données se décomposent en 3 éléments :

* Dimension (un entier n compris entre 12 et 100)
* Matrice décrivant w la fonction de poids définie entre les équipements
* Matrice décrivant d la distance entre les emplacements

On a aussi la solution optimale pour certains jeu de données.

On considérera les matrices de poids et de distance comme étant symétriques, ce qui est le cas pour le jeu de donnée que l’on utilise.

# Choix de nommage et définitions

Dans ce projet j’appelle configuration ou solution, les différentes façons possibles d’affecter n équipements sur n emplacements. Je représente une configuration par un tableau d’entier. Par exemple {2,3,1} indique que la machine 2 est en position 1, que la machine 3 est en position 2 et que la machine 1 est en position 3.

Une configuration est correcte si et seulement si chaque machine y est présente une seule fois et le numéro de chaque machine est bien compris entre 1 et n (inclus) ;

Soit C l’ensemble des configurations correctes, soit

* \*

# Voisinages

On commence par prendre une transformation élémentaire simple : la permutation de l’emplacement de deux équipements.

Ce choix de voisinage me semble intéressant car il permet d’assurer que les voisins d’un configuration correcte sont aussi corrects.

Le nombre de voisins est , il est possible que cela soit un nombre trop important lorsque n est grand : .

On note , pour la méthode Tabou, que l’opération inverse d’une permutation correspond a la même permutation.

# Paramètres initiaux des algorithmes

Les paramètre pour l’algorithme de recuit simulé sont les suivants :

* NB\_Steps : Nombre d’itérations (steps)
* T0 : Température initiale
* Mu  : valeur par laquelle la température est multipliée a chaque itération
* P  : probabilité servant à définir la température initiale

Je calcule T0 grâce a la valeur P donnée. Je sélectionne 3 configurations aléatoire, je prends la plus grosse différence de fitness entre une configuration et ses voisins. Enfin je calcule la température initiale de sorte a ce que la pire différence de fitness ait une probabilité p d’être acceptée par l’algorithme.

Les paramètres pour la méthode Tabou sont les suivants :

* NB\_Steps : Nombre d’itérations (steps)
* TabouLength : taille de la liste Tabou

La solution initiale est sélectionnée aléatoirement.

# Implémentation

J’ai décidé de coder le projet d’une manière à pouvoir facilement changer les opérations élémentaires et le landscape du modèle (grâce aux interfaces Landscape et ElementaryOperation).

J’ai implémenté la marche aléatoire, le recuit simulé et la méthode Tabou.

## Vérification de l’implémentation

Pour vérifier le comportement correct, j’ai testé les algorithmes sur un problème très simple, de dimension 3

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Poids(weight)   |  |  |  | | --- | --- | --- | | 0 | 1 | 2 | | 1 | 0 | 1 | | 2 | 1 | 0 | | Distances (dist)   |  |  |  | | --- | --- | --- | | 0 | 3 | 1 | | 3 | 0 | 2 | | 1 | 2 | 0 | |

Ce qui correspond à ce schéma de l’ensemble des configurations et de leur voisinage :

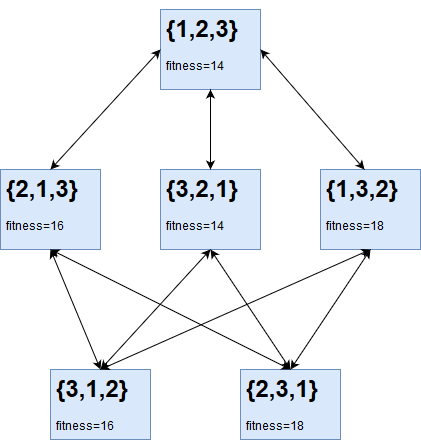


Figure : schéma de l'instance de test

Dans ces tests les algorithmes commenceront sur la configuration (1,2,3).

Le détail de ces tests se trouve sur classeursExcel/Verification\_implementation.xlsx

### Recuit simulé

Ainsi voici les résultats pour l’algorithme de recuit simulé avec les paramètres mu=0.8 , p=0.7 et NB\_Steps =30 :

|  |  |  |
| --- | --- | --- |
| Step | Configuration | Fitness |
| 0 | Config{ 1, 2, 3 ) | 14 |
| 1 | Config{ 1, 3, 2 ) | 18 |
| 2 | Config{ 3, 1, 2 ) | 16 |
| 3 | Config{ 2, 1, 3 ) | 16 |
| 4 | Config{ 2, 3, 1 ) | 18 |
| 5 | Config{ 1, 3, 2 ) | 18 |
| 6 | Config{ 1, 2, 3 ) | 14 |
| 7 | Config{ 2, 1, 3 ) | 16 |
| 8 | Config{ 3, 1, 2 ) | 16 |
| 9 | Config{ 3, 2, 1 ) | 14 |
| 10 | Config{ 3, 2, 1 ) | 14 |
| 11 | Config{ 3, 2, 1 ) | 14 |
| 12 | Config{ 3, 2, 1 ) | 14 |
| 13 | Config{ 3, 2, 1 ) | 14 |
| 14 | Config{ 3, 2, 1 ) | 14 |
|  | … |  |

On observe que l’on parcoure des configurations avec une mauvaise fitness lors des premières itérations et que rapidement l’algorithme se stabilise sur les configurations avec les meilleures fitness (1,2,3) et (3,2,1).

C’est le déroulement attendu du recuit simulé alors je considère que mon implémentation est correcte.

### Méthode Tabou

Voici les résultats de la méthode Tabou sur l’instance de test avec TabouLength=1.

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Configuration | Fitness | Tabous |
| 0 | Config{ 1, 2, 3 ) | 14 |  |
| 1 | Config{ 3, 2, 1 ) | 14 | (0-2) |
| 2 | Config{ 3, 1, 2 ) | 16 | (1-2) |
| 3 | Config{ 2, 1, 3 ) | 16 | (0-2) |
| 4 | Config{ 1, 2, 3 ) | 14 | (0-2) |
| 5 | Config{ 2, 1, 3 ) | 16 | (0-1) |
| 6 | Config{ 3, 1, 2 ) | 16 | (0-2) |
| 7 | Config{ 3, 2, 1 ) | 14 | (0-2) |
| 8 | Config{ 3, 1, 2 ) | 16 | (1-2) |
| 9 | Config{ 2, 1, 3 ) | 16 | (0-2) |
| 10 | Config{ 1, 2, 3 ) | 14 | (0-2) |
| 11 | Config{ 2, 1, 3 ) | 16 | (0-1) |
| 12 | Config{ 3, 1, 2 ) | 16 | (0-2) |
| 13 | Config{ 3, 2, 1 ) | 14 | (0-2) |
| 14 | Config{ 3, 1, 2 ) | 16 | (1-2) |
| 15 | Config{ 2, 1, 3 ) | 16 | (0-2) |
| 16 | Config{ 1, 2, 3 ) | 14 | (0-2) |
|  | … |  |  |

On observe qu’à chaque itération l’algorithme se dirige bien au voisin non interdit par la liste Tabou qui as la meilleure fitness. On observe que l’algorithme se retrouve rapidement dans une boucle.

C’est le comportement attendu de la méthode Tabou donc je considère mon implémentation correcte.