Problème du voyageur de commerce

Document Organique

TRANG Hoang Phong Vu
AUGUSTO Rold
PAUL Hugo
SUGANATHASIVAM Vathanalakshan

12/11/2020

Table des matières

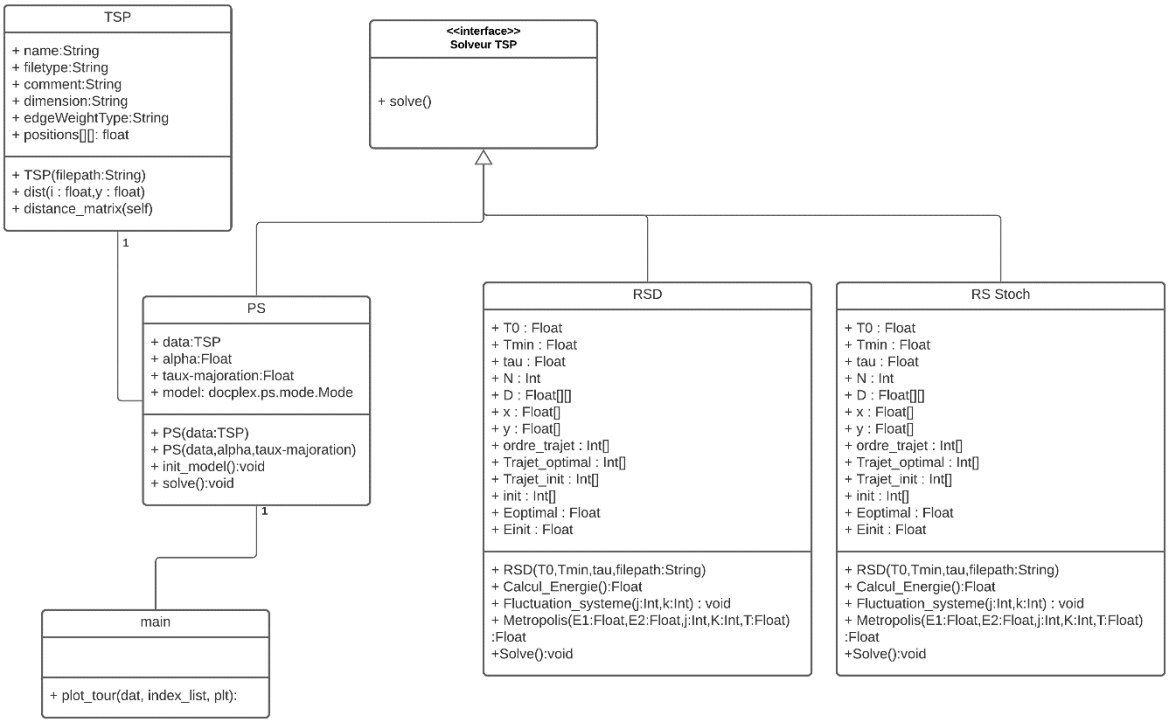
Introduction

Le problème du voyageur de commerce a été traité durant ce projet en utilisant CPLEX et Python principalement. Ce document organique montre la structure de code et permet d'expliquer la signifiance de chacune de classes, variables et méthodes. Un diagramme UML est aussi présent pour comprendre la structure globale du projet.

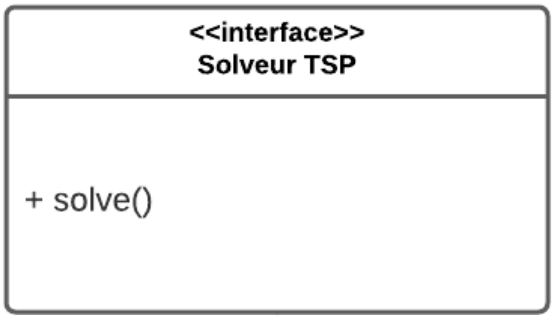
Read Me

```
Pré-requis pour la partie Cplex :
    * Python 3.7
    * CPLEX Studio
Le démarche pour compiler :
    * Activer l'environnement tsp_venv
    * Initialiser cplex au python:
        * Lancer setup.py dans la répertoire /chemin/de/CPLEX/cplex/python/3.7/PLATFORM/:
            python setup.py install
    * Lancer main.py
```

Diagramme UML



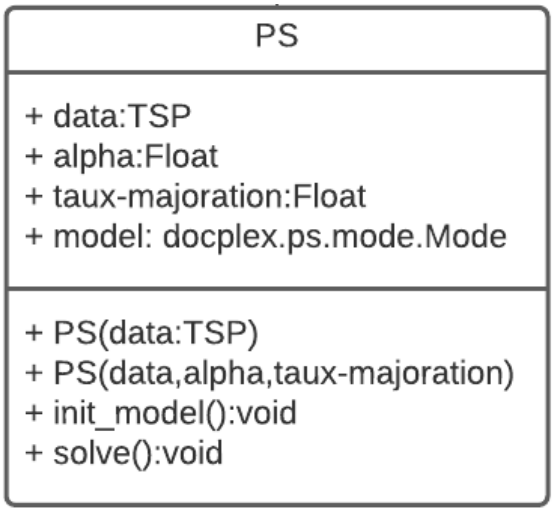
Classes Expliquées



L'interface permettant aux classes de CPLEX et Recuit Simulé d'implémenter la fonction solve.

CPLEX

Programmation Déterministe et Stocastique



La classe permettant de calculer le problème en utilisant CPLEX.
Variables :

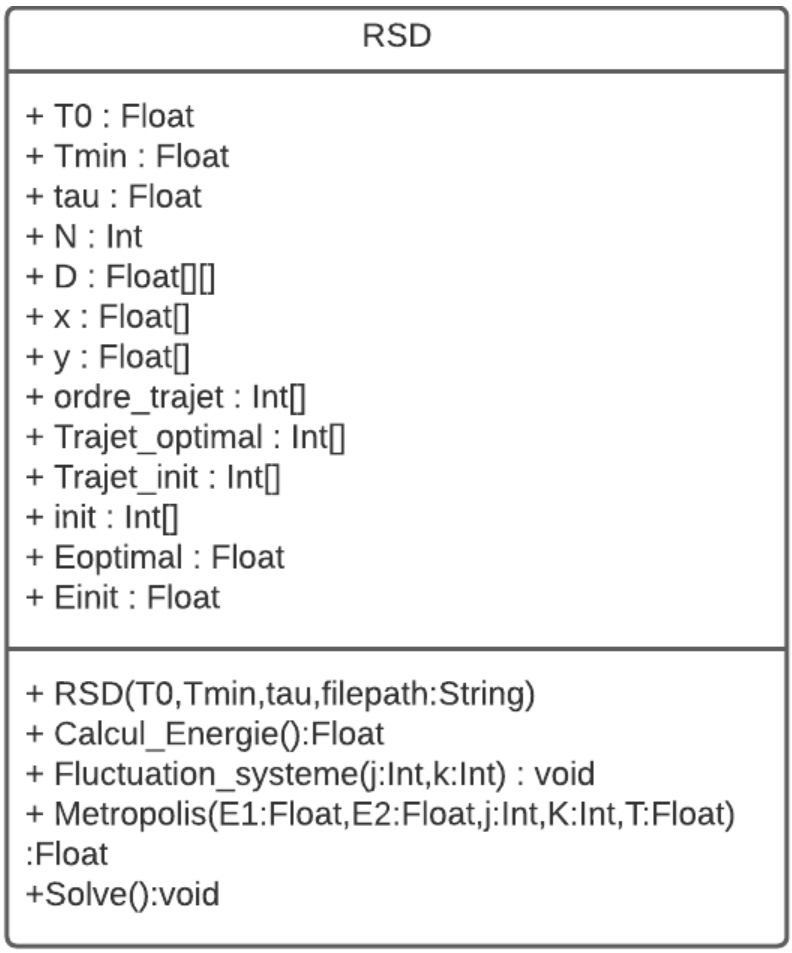
- Data : Les données en format.
- Alpha : La valeur de l'α.
- Taux-majoration : Le taux de majoration.
- Model : Le format permettant de connecter python a CPLEX.

Méthodes :

- PS () : Le constructeur pour la version déterministique.
- PS (data, alpha, taux-majoration) : Le constructeur pour la version stochastique.
- init_model () : Le modelé représentant les contraintes et la fonction objective.
- solve () : La fonction qui calcule la solution du problème.

Recuit Simulé

Recuit Simulé Déterministe



La Classe permettant de calculer le recuit simulé de façon déterministe

Variables :

- T0 : Température initial.
- Tmin : Température minimal.
- tau : Le tau de la loi de la décroissance de température.
- N : Nombre de points/ville.
- D : La matrice représentant la distance entre chaque ville.
- x : Coordonnée X de chaque point.
- y : Coordonnée Y de chaque point.
- ordre_trajet : Tableau représentant le trajet sans le retour au point de départ.
- Trajet_optimal : Tableau représentant le trajet final avec le retour au point de départ.
- Trajet_init : Tableau représentant le trajet de départ avec le retour au point de départ.
- init : Tableau représentant le trajet de départ sans le retour au point de départ.
- Eoptimal : Distance optimal parcouru.
- Einit : Distance de départ parcouru.

Méthodes :

- RSD(T0,Tmin,tau,filepath:String) : Le constructeur.

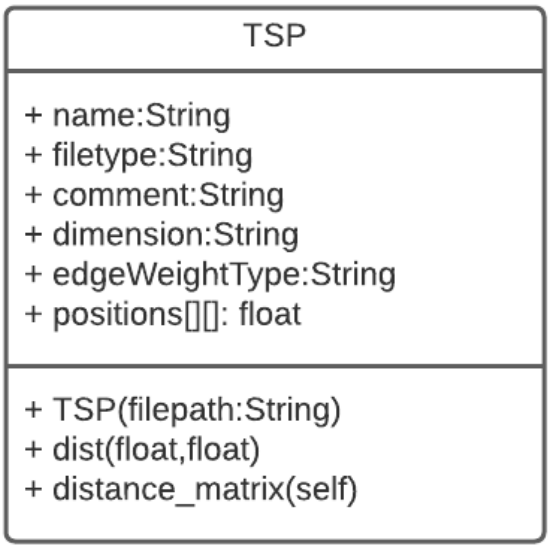
- Calcul_Energie(): Calcule la distance pour le trajet actuelle.
- Fluctuation_systeme(j:Int,k:Int) : Modifie le trajet .
- Metropolis(E1:Float,E2:Float,j:Int,K:Int,T:Float) : Algorithme permettant de comparer l'ancien trajet avec le trajet post-fluctuation et conserve la distance optimale entre les deux trajets.
- Solve():La fonction qui calcule la solution du problème en mode recuit simulé et l'affiche les résultat et graphes.

Recuit Simulé Stochastique

TODO

Classes Annexes

TSP



Une classe permettant de lire un fichier TSP et stocker les données y présentes. Elle permet aussi de calculer la matrice de distance entre les points.

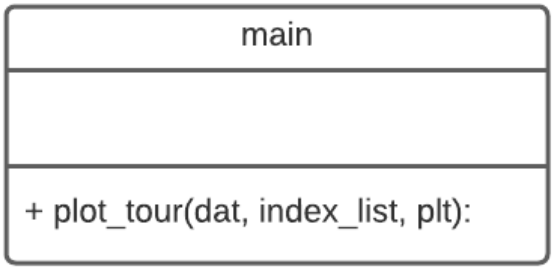
Variables :

- name : Les données en format.
- filetype : Le type du fichier.
- comment : Le commentaire présent dans fichier TSP.
- dimension : Nombre de points.
- edgeWeightType : Le type de points.
- positions [][] : Tableau représentant les points des données.

Méthodes :

- TSP () : Le constructeur pour la classe.
- dist(float,float) : Fonction qui calcule les distances entre deux points.
- distance_matrix(self) : Fonction qui calcule la distance entre tous les points et forme une matrice.

Main



Fonction main qui calcule le problème avec CPLEX soit en mode stochastique ou déterministe. Ensuite elle s'occupe d'afficher les données dans un plot.

Méthodes :

- plot_tour(dat,index_list,plt) : Dessine le graph en utilisant les données(dat) et listes de points(index_list).