Projet SY10 sur l'optimisation et la recherche de chemin pour l'accès à l'eau en Éthiopie

---

> **Tip**
>
> Ce projet est sur GitHub (https://github.com/teo-bou/SY10) et nous y ferons peut-être des modifications donc si vous voulez accéder à la dernière version, n'hésitez pas à y aller. Par ailleurs, si vous avez quelques questions que ce soit, n'hésitez pas à nous recontacter

> **Warning**
>
> Certaines bibliothèques Python ont été utilisées pour ce projet. Pour les installer, utilisez la commande `pip install -r requirements.txt`

Le projet

Pour commencer :

Le fichier d'interface principal, où les cartes, les villages, les sources peuvent être rentrés est le fichier `/Projet-Eau/recherche_chemin_eau.ipynb`
C'est un fichier python notebook et nous vous recommandons vs-code pour l'ouvrir.

- **Définition de la carte :**

1. Entrez le type de terrain et l'accessibilité de la zone géographique visée
    1. Pour cela, utilisez les `Classe_classification type_terrain et accessibilite` et leurs méthode `.v()` (cf. Architecture du projet)
        1. ex : `type_terr = type_terrain.v(0.2, 1, 0.3)` , ici on évalue le terrain comme étant "peu escarpe" avec un degré de 0.2, "moyen" avec un degré de 1 et "escarpe" avec un degré de 0.3
2. Créez la carte selon ce modèle
    1. `carte = Carte(nom de l'image dans le dossier carte, date (jj/mm), type_terrain=type_terr (défini auparavant), accessibilite=access (définie auparavant), resized_factor = facteur d'échelle permettant d'accélérer en approximant les calculs dans la recherche de chemin plus tard)`
    2. Vous pouvez affichez la carte et sa vue en 3D avec `print(carte)`

- **Définitions des villages et des sources**

1. Créez une liste contenant tous les villages
    1. Un village est défini comme tel : `Village(carte, x à l'échelle réelle, y à l'échelle réelle, IFT représentant le nombre d'habitants, ressenti_conflits.v(..,..,..,..),ressenti_enthousiasme.v(..,..,..), {"hopital": .., "ecole": .., "gouvernement": ..} (nombre d'infrastructures), nom du village)`
2. Créer une liste contenant toutes les sources
    1. Une source est définie comme telle : `Source(carte, x à l'échelle réelle, y à l'échelle réelle, couleur_eau.v(.., ..), IFT représentant le débit de la source en L/s , odeur_eau.v(.., .., ..), nom de la source)`

- **Score de faisabilité**
    - `faisabilité(carte, liste_villages, liste_sources, show= permet d'afficher des logs, defuz= permet d'avoir la sortie défuzzifiée)`
- **Tracé sur la carte et calcul de chemin**

- `tracer_carte(carte, liste_villages, liste_sources, show = permet d'afficher les logs des étapes intermédiaires)`

Architecture du projet :

- `utilities.py` contient la boîte à outils pour le flou que nous avons recréé. C'est ici que vous retrouverez comment le concept d'un IFT, d'un SIF (que l'on a nommé Table)... on été implémentés. Elle a été conçue de manière à être assez générale et réutilisée pour d'autres projets.
- Le dossier `/Projet-Eau` est plus spécifique pour notre projet. Vous y retrouverez :
  - `Classes.py` contenant les définitions de nos différentes classes floues et leurs partitions
    - Vous pouvez utiliser `Classe.plot()` pour avoir une représentation graphique de la classe
    - les objets de type `Classe_classification` sont utilisés lorsque l'entrée est entrée directement fuzzifiée en floue et sont utiles pour avoir les différentes classes que la sortie peut avoir
      - On peut créer un dictionnaire de représentation d'une instance avec la fonction `Classe_classification.v(~ les différentes valeurs des fonctions d'appartenances)`
  - `Rules.py` contient les définitions des différents systèmes flous
    - Les systèmes flous à 3 dimensions sont représentés par des objets de type `Table_mult(classe différenciante de quelle table utiliser, les tables séparées par une virgule, meaning = le sens du système flou de manière à garder de l'intelligibilité)`
  - `flou-import.py` qui permet d'avoir accès à `utilities.py` même s'ils ne sont pas dans le même dossier
  - `projet_srcs.py` permet aussi d'importer tous les outils nécessaires pour `recherche_chemin_eau.ipynb`
  - `Calcul_flou.py` regroupe toutes les cascades de SIFs ainsi que la majorité des différentes fonctions utilisées
    - `faisabilité(carte, liste_villages, liste_sources, show= permet d'afficher des logs, defuz= permet d'avoir la sortie défuzzifiée)` permet de calculer le score global de faisabilité
    - `tracer_carte(carte, liste_villages, liste_sources, show = permet d'afficher les logs des étapes intermédiaires)` permet d'afficher la carte ayant les chemins dessus
  - `Objets.py` rassemble les définitions des différents objets utiles à notre projet (Village, Source, Carte...)
  - `Astar.py` est l'implémentation de l'algorithme A* pour notre projet, algorithme en charge de la recherche du meilleur chemin entre deux points
  - `matching.py` permet l'association des villages et des sources entre eux grâce à leur score calculé dans `Calcul_flou.py`
  - `/cartes/` dossier où sont placés les images définissants les cartes
    - Les cartes que nous utilisons proviennent de ce site : https://portal.opentopography.org/raster?opentopoID=OTSDEM.032021.4326.3
  - `/SIFS/` dossier où sont placés les tableaux de règles, ainsi qu'un script permettant de les corriger