Devoir 2 - Gestion de points de distributions

Remise le 30/03/25 (avant minuit) sur Moodle pour tous les groupes.

Consignes

- Le devoir doit être fait par **groupe de 2 au maximum**. Il est fortement recommandé d'être 2.
- Lors de votre soumission sur Moodle, donnez votre rapport en format **PDF** ainsi que vos fichiers de code à la racine d'un seul dossier compressé nommé (`matricule1_matricule2_Devoir2.zip`).
- Indiquez vos noms et matricules en commentaires au dessus des fichiers .py soumis.
- Toutes les consignes générales du cours (interdiction de plagiat, etc.) s'appliquent pour ce devoir.
- Il est permis (et encouragé) de discuter de vos pistes de solution avec les autres groupes. Par contre, il est formellement interdit de reprendre le code d'un autre groupe ou de copier un code déjà existant (StackOverflow ou autre). Tout cas de plagiat sera sanctionné de la note minimale pour le devoir.

Énoncé du devoir

Votre analyse de marché est un succès, Jia et vous ne tardez pas à crouler sous les demandes de partenariats. C'est ainsi que vous trouver votre prochaine mission chez l'entreprise TONDA. Une partie des communautés identifiées partageraient en effet un goût prononcé pour les produits de la mer. Malheureusement, l'exportation de ces denrées chez eux nécessite un transport délicat et reste coûteux pour l'entreprise. Ils font donc appel à vous pour trouver une façon harmonieuse de distribuer leurs produits dans plusieurs centres de vente depuis les quelques lieux de distributions à leur disposition grâce aux camions de l'entreprise. Vous devrez donc engendrer le maximum de revenu en reliant le plus de points de vente possible par un traçage, noté T, respectant des contraintes assez particulières.

Portée des livraisons

Chaque région comportant un centre de distribution de l'entreprise est représentée par un graphe connecté, non orienté et sans cycle $G = (V, E)$, avec V l'ensemble des nœuds de taille N et E l'ensemble des arêtes de taille M. Chaque nœud indique un point de passage potentiel pour les camions de livraison et les arêtes sont les routes les liant entre eux. Votre traçage T devra indiquer les routes empruntées par les transporteurs, on dénotera donc $T \subseteq E$. Il existe également deux types de points d'intérêts par région :

1. Un lieu désignant le centre de distribution d'où les camions doivent commencer leur voyage, il sera toujours dénoté comme le premier nœud.
2. Un ensemble de points de vente $P \subseteq V$ identifiant les lieux où l'entreprise engendre un revenu à la livraison de ses produits. On notera donc ce revenu $r_v, v \in V$ avec $r_v = 0, v \in V \setminus P$.

Notez que le centre de distribution fait toujours partie de l'ensemble P car des clients peuvent aussi s'en servir comme point de vente. Une partie du revenu sera donc toujours engendrée par le centre. Notez également que le revenu n'est engendré que par le premier passage dans un lieu, les cycles n'ont donc aucun intérêt dans vos solutions.

Déploiement des livraisons depuis le centre

Pour que votre traçage T soit valide, aucune route ne doit être déconnectée du centre de distribution. Ainsi, si l'on dénote $U_T \subseteq V$ l'ensemble des lieux atteint par le traçage T, pour chaque lieu $u \in U_T$, il doit exister un chemin $\mathscr{P}_u \subseteq U_T$ composé uniquement d'autres lieux le reliant au centre de distribution et en passant uniquement par les routes dans T. Vous pouvez aussi considérer que le sous-graphe $G' = (U_T, T)$ doit être un composant connexe et que le centre de distribution est forcément compris dans l'ensemble U_T.

Budget des livraisons

L'entreprise attribue un budget B à chaque région, ainsi chaque route $e \in E$ possède un coût associé c_e représentant la distance et la difficulté de circulation entre deux lieux. Vos solutions T devront alors se plier au budget décidé par l'entreprise selon la contrainte suivante :

$$\sum_{e \in T} c_e \leq B \tag{1}$$

Limitation de distance

Les conditions particulières dans lesquelles vos produits doivent être maintenus vos produits dans les camions imposent une limitation de la distance entre le centre de distribution et les lieux reliés par votre traçage. L'entreprise définira, pour chaque région, une distance maximale H en nombre de lieux entre le centre de distribution et un lieu désigné. Chaque lieu que rejoindra votre traçage T devra donc se trouver dans cette limite. On peut donc formaliser cette contrainte, à l'aide de U_T et des chemins $\mathscr{P}_u, u \in U_T$ définis plus haut, de cette façon :

$$|\mathscr{P}_u| \leq H, u \in U_T \tag{2}$$

Votre mission

Votre mission sera donc de trouver un traçage maximisant le revenu de l'entreprise selon la formule suivante :

$$\sum_{u \in U_T} r_u \tag{3}$$

Le coût des routes, étant attribué à un budget maximal dans une contrainte, n'est donc pas pris en compte dans la fonction objectif. À vous donc de trouver le meilleur traçage T possible, reliant les lieux aux meilleurs profits et en plus grand nombre, tout en respectant les contraintes énoncées plus haut.

Format des instances

Chaque fichier d'instance du dossier `instances`, à l'exception de l'instance triviale servant d'exemple, est nommé selon une lettre d'identification et les paramètres qui la compose selon le format suivant : `<instance_letter>_<N>_<M>_<P>_<B>_<H>.txt`. Chaque groupe de lieux associé à un centre de distribution est inscrit dans un fichier d'instance de $M + P + 1$ lignes correspondant au format suivant.

```
1 N M P B H
2 E <e_1_1> <e_1_2> <c_1>
```

```
E <e_2_1> <a_2_2> <c_2>
...
E <e_M_1> <a_M_2> <c_M>
PV <n_1> <r_1>
PV <n_2> <r_2>
...
PV <n_P> <r_P>
```

La première ligne de l'instance indique les variables globales du problème, soit le nombre de lieux, le nombre de routes, le nombre de lieux propice à l'installation d'un commerce, le budget maximal et le nombre de routes maximum entre le centre de distribution et les lieux. En suivant, les M prochaines lignes indiquent les routes entre deux lieux et le coût associé. Enfin, les P dernières lignes indiquent les lieux profitables et le revenu engendré lorsqu'ils sont liés au centre de distribution. Ainsi, dans l'instance supplémentaire `trivial.txt` affichée ci-dessous, le budget maximal est de 93 (ligne 1), les lieux 6 et 17 sont liés par une route de coût 7 (ligne 9) et le lieu 22 génère un profit de 52, car il apparait dans la liste des nœuds PV (ligne 27). Une représentation graphique de l'instance est proposée dans la figure 1. Ici, le nœud rouge indique le centre de distribution, les nœuds orange indiquent les lieux exploitables, tous les autres lieux sont représentés en bleu. Les routes entre chaque lieu sont tracées par les arêtes du graphe.

```
22 24 3 93 5
E 1 2 7
E 2 8 8
E 2 19 5
E 3 5 1
E 3 17 6
E 3 11 7
E 4 5 8
E 6 17 7
E 7 9 3
E 7 20 3
E 9 20 5
E 10 22 6
E 11 5 9
E 12 9 5
E 13 17 1
E 14 20 7
E 15 1 10
E 15 7 10
E 16 20 8
E 17 21 5
E 18 19 2
E 18 21 10
E 20 22 2
E 21 22 2
PV 1 88
PV 7 43
PV 22 52
```

Format des solutions

Il vous est demandé de fournir pour chaque configuration un fichier de solution. Le fichier doit contenir $|T|+1$ lignes. La première ligne indique le nombre de routes dans votre solution ($|T|$), et les lignes suivantes indiquent les routes du traçage. Une fonction utilitaire `save_solution()` vous est fournie pour générer ce

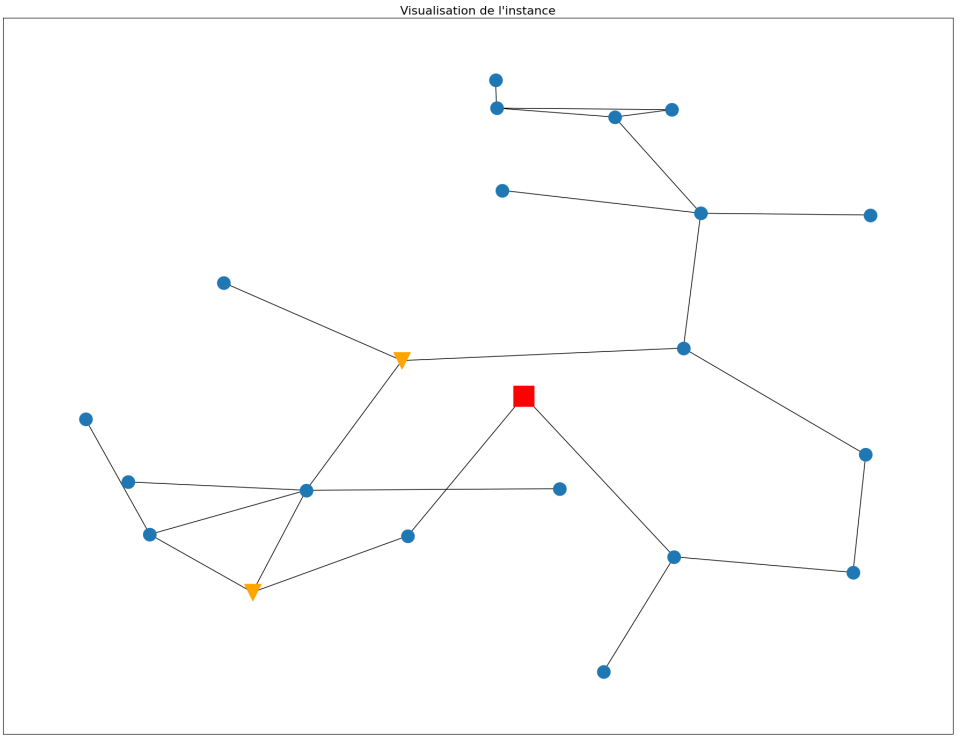

FIGURE 1 – Représentation graphique de l'instance `trivial`.

fichier.

```
<|T|>
<e_1_1> <e_1_2>
<e_2_1> <e_2_2>
...
<e_T_1> <e_T_2>

```

Exemple illustratif

À titre d'exemple, voici la solution qu'un solveur naïf pourrait retourner sur l'instance `trivial.txt`. Ici, le solveur ajoute des routes depuis le centre de distribution tant que les contraintes peuvent être respectées.

```
8
8 2
1 15
1 2
2 19
9 7
20 7
18 19
7 15
```

Une illustration de cette solution est donnée à la Figure 2. Sur cette figure, vous trouverez également le budget utilisé et le revenu engendré par le traçage. La figure montre le graphe de l'instance avec des nœuds de formes différentes selon la nature du lieu et la partie des routes présente dans le traçage soumis en solution. Le carré rouge représente le centre de distribution, les triangles orange sont les lieux de vente et le reste des lieux sont les ronds bleus. Cette représentation identifie chaque chemin du traçage complet pour leur attribuer un marquage permettant de les distinguer. À titre de comparaison, vous pourrez voir la visualisation d'une meilleure solution sur la Figure 3. Remarquez également l'évolution du revenu et de la consommation du budget entre ces deux solutions. Des fonctions utilitaires vous sont données pour générer ces figures.

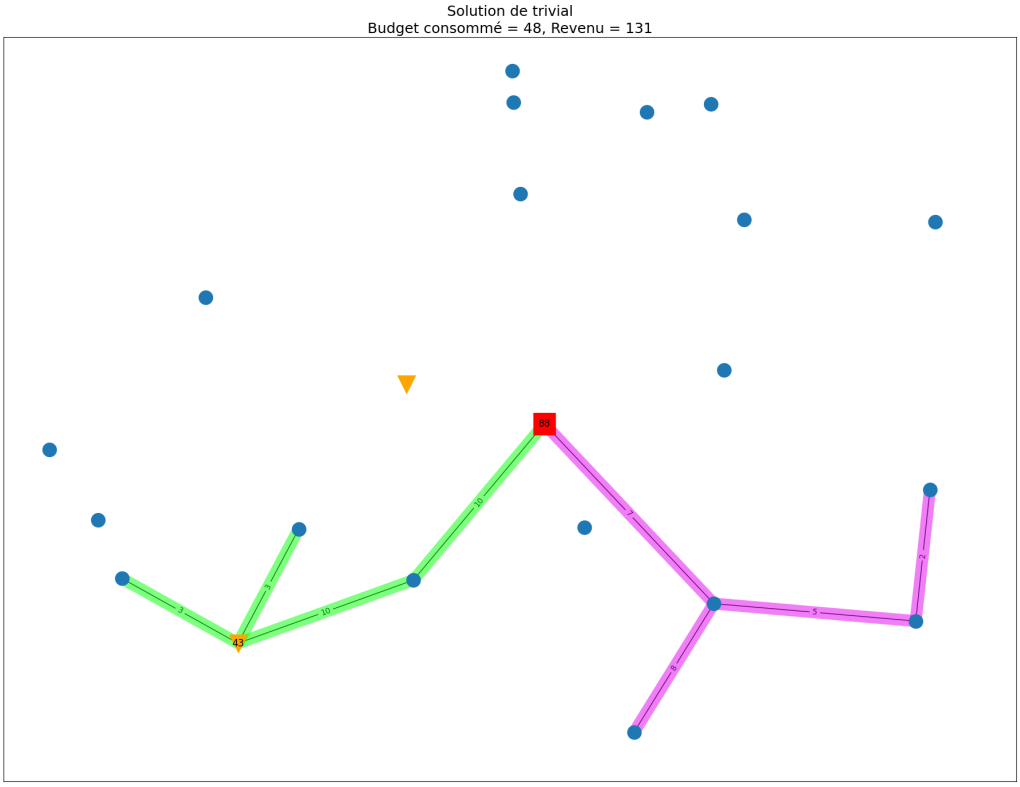

FIGURE 2 – Visualisation de la solution illustrative pour l'instance `trivial`.

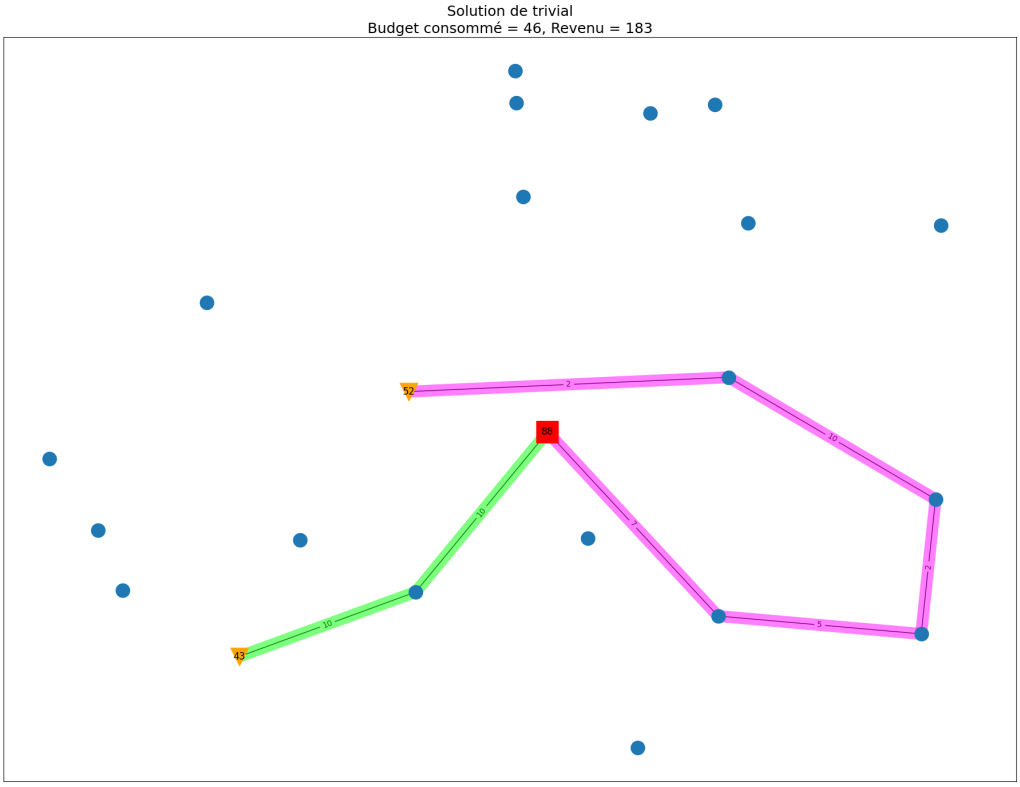

FIGURE 3 – Visualisation d'une solution plus avancée pour l'instance `trivial`.

Implémentation

Vous avez à votre disposition un projet python. Plusieurs fichiers vous sont fournis :

- `requirements.txt` qui contient les librairies nécessaires au projet.
- `utils.py` qui implémente la classe `Instance` pour lire les instances, sauvegarder les solutions, les valider et les visualiser.
- `main.py` vous permettant d'exécuter votre code sur une instance donnée. Ce programme enregistre également votre meilleure solution dans un fichier au format texte et sous la forme d'une image.
- `solver_naive.py` : implémentation d'un solveur naïf ajoutant toutes les routes après le centre tant que les contraintes le permettent.
- `solver_advanced.py` : implémentation de votre méthode de résolution qui sera évaluée pour ce devoir.
- `autograder.py` : un programme vous permettant de facilement calculer votre note compte tenu de vos performances sur les instances fournies.

Notez bien qu'un vérificateur de solutions est disponible. Vous êtes également libres de **rajouter d'autres fichiers au besoin**. Au total, 6 instances sont disponibles dans les fichiers `instances/<instance>.txt` en plus de l'instance triviale `instances/trivial.txt`. Il vous est demandé de produire les fichiers de solution `solutions/<instance>.txt` pour chacune de ces instances. Pour vérifier que tout fonctionne bien, vous pouvez exécuter le solveur naïf comme suit.

```
python3 main.py --agent=naive --infile=./instances/trivial.txt
```

Un fichier `solutions/trivial.txt` décrit plus haut et une image `visualization_trivial.png` du même format que la Figure 2 seront générés.

Un environnement virtuel sous **Python 3.11** est recommandé afin d'installer les librairies du projet grâce à la commande suivante.

```
pip3 install -r requirements.txt
```

Production à réaliser

Vous devez compléter le fichier `solver_advanced.py` avec votre méthode de résolution. Au minimum, votre solveur doit contenir un algorithme de recherche locale. C'est-à-dire que votre algorithme va partir d'une solution complète et l'améliorer petit à petit via des mouvements locaux. Réfléchissez bien à la définition de votre espace de recherche, de votre voisinage, de la fonction de sélection et d'évaluation. Vous devez également intégrer une métaheuristique de votre choix pour vous s'échapper des minimums locaux (autre qu'un simple redémarrage aléatoire). Vous êtes ensuite libres d'apporter n'importe quelle modification pour améliorer les performances de votre solveur. L'utilisation de librairies python externe est autorisée, mais doivent être utilisée avec parcimonie, si une librairie remplace entièrement les attendus de base, votre code ne sera pas considéré comme remplissant les exigences du devoir. Une fois construit, votre solveur pourra ensuite être appelé comme suit.

```
python3 main.py --agent=advanced --infile=./instances/trivial_1.txt
```

L'option `-no-viz` peut être ajoutée afin de ne pas avoir à générer la visualisation à chaque fois.

Un rapport **succinct** (2-3 pages de contenu, sans compter la page de garde, figures, et références) doit également être fourni. Dans ce dernier, vous devez présenter votre algorithme de résolution, vos choix de conceptions, vos analyses de complexité, et toute autre information que vous jugez pertinente. À titre d'exemple, **il est attendu que vous analysiez la complexité des fonctions principales de votre algorithme**, comme par exemple la sélection d'un voisin en fonction de la taille du voisinage. Reportez-y également les résultats obtenus pour les différentes instances. Finalement, vous devez fournir un dossier `solutions` avec vos solutions au format telles qu'écrites plus haut et en respectant la structure `solutions/<instance_name>.txt`.

Critères d'évaluation

L'évaluation portera sur la qualité du rapport et du code fournis, ainsi que sur les performances de votre solveur sur les différentes instances. Concrètement, la répartition des points (sur 20) est la suivante :

— **10 points sur 20** sont attribués à l'évaluation des solutions fournies par votre solveur. Parmi ces instances, six d'entre elles vous sont fournies et une dernière restera cachée pour vérifier les capacités de généralisation de votre solveur. Les scores que vous obtiendrez devront se situer entre les valeurs (**LB**) et (**UB**) du tableau. Obtenir la valeur **UB** vous permettra d'obtenir 100% de la note pour cette instance, obtenir une valeur inférieure ou égale à **LB** ne vous fera gagner aucun point (0%). La perte des points par rapport à **UB** est linéaire entre les deux bornes. A titre d'exemple, pour l'instance D, j'ai obtenu une instance valide de coût 3682, ma note pour cette instance est de $(1 \times \frac{3682-LB}{UB-LB}) \approx 0.85/1$.

Le tableau ci-dessous indique toutes les caractéristiques des instances ainsi que certaines informa-

tions de l'instance cachée. La dernière colonne indique le temps de résolution maximal accordé à chaque instance.

Instance	Pondération	N	M	P	B	H	LB	UB	Temps max.
A	1/10	75	150	13	85	12	412	694	5min
B	1/10	100	200	25	103	12	945	1225	5min
C	1/10	500	625	83	342	15	2181	2969	5min
D	1/10	500	1000	125	272	25	2415	3910	5min
E	2/10	500	2500	125	136	5	171	367	10min
F	2/10	500	12500	125	345	15	179	648	10min
X	2/10	500	12500	250	68	5	?	?	10min

TABLE 1 – Résultats attendus pour la notation

— **10 points sur 20** sont attribués à l'évaluation de votre rapport. Sans être exhaustif, les éléments indispensables pour obtenir une bonne note sont :

1. Une formalisation et une analyse de **votre espace de recherche** (p.ex., est-il connecté?).
2. **Une quantification chiffrée et/ou argumentée** de l'impact des mécaniques implémentées (p.ex., vous échappez-vous bien des extrema locaux? Si oui, prouvez-le, sinon, montrez *formellement* pourquoi (Vous avez une stratégie qui permet de réduire la complexité d'une routine? Prouvez que ce que vous faites est utile avec des *mesures de temps*, de *mémoire* ou en *donnant une notation asymptotique*).
3. **Privilégiez** graphes, tableaux et formules pour transmettre l'information de manière concise.
4. **Évitez à tout prix** les explications de nature qualitatives, rapporter qu'un algorithme "va plus vite", qu'il est "plus performant" ou "donne de meilleures solutions" est flou et incomplet. À l'inverse, arriver à le prouver ou à le quantifier expérimentalement est nécessaire dans un rendu scientifique.

Une fois à la racine de votre projet, vous pouvez calculer vos scores automatiquement avec :

```
python3 autograder.py
```

⚠**Les solutions des instances doivent se trouver dans** `./solutions/` **et respecter la nomenclature demandée. N'oubliez pas de générer à nouveau vos solutions pour évaluer une nouvelle méthode ou un changement.**

⚠**Il est attendu que vos algorithmes retournent une solution et des valeurs correctes. Par exemple, il est interdit de modifier artificiellement le nombre de nœuds, arêtes ou autre. Un algorithme retournant une solution non cohérente est susceptible de ne recevoir aucun point.**

Conseils

Voici quelques conseils pour le mener le devoir à bien :

1. Tirez le meilleur parti des séances de laboratoire encadrées afin de demander des conseils.
2. Inspirez vous des techniques vues au cours, et ajoutez-y vos propres idées.
3. Procédez par étape. Réfléchissez d'abord sur papier à une approche, implémentez une recherche locale simple, puis améliorez-la avec vos différentes idées.

4. Tenez compte du fait que l'exécution et le paramétrage de vos algorithmes peut demander un temps considérable. Dès lors, ne vous prenez pas à la dernière minute pour réaliser vos expériences.

Remise

Vous remettrez sur Moodle une archive zip nommée `matricule1_matricule2_Devoir2.zip` contenant :

— Votre code commenté au complet.
— Un dossier `solutions` avec vos solutions au format attendu par TONDA (décrit plus haut) respectant la structure `solutions/<instance_name>.txt`.
— Votre rapport de présentation de votre méthode et de vos résultats respectant les modalités énoncées plus haut.