MANUEL TECHNIQUE

I- Présentation de l'IA implémentée

L'IA implémentée est un système expert utilisant la propagation arrière. Cela signifie que l'on va chercher à savoir si notre diagnostic est vrai selon les conditions de départs. On va donc fournir au système expert une liste de diagnostiques, ce dernier va regarder selon les règles et d'après les conditions initiales quels diagnostiques sont valide.

Dans l'IA implémentée, les règles sont les suivantes :

S'il y a moins d'une école pour 500 élevés ALORS il faut ajouter une école.
S'il y a au moins d'une école pour 500 élevés ALORS il y a suffisamment d'écoles.
S'il y a moins d'un magasin pour 2000 habitant ALORS il faut ajouter un magasin.
S'il y a au moins un magasin pour 2000 habitant ALORS il y a suffisamment de magasins.
S'il y a moins d'une gendarmerie pour 20000 habitants ALORS il faut ajouter une gendarmerie.
S'il y a au moins une gendarmerie pour 2000 habitants ALORS il y a suffisamment de gendarmeries.
S'il y a moins de une caserne de pompier pour 20000 habitants ALORS il faut ajouter un caserne de pompiers.
S'il y a au moins une caserne de pompier pour 20000 habitants ALORS il y a suffisamment de casernes de pompiers.
S'il y a moins de un centre médical pour 500 habitants ALORS il faut ajouter un centre médical.
S'il y a au moins un centre médical pour 500 habitants ALORS il y a suffisamment de centre médicaux.
S'il y a moins de une pharmacie pour 1000 habitants ALORS il faut ajouter une pharmacie.
S'il y a au moins une pharmacie pour 1000 habitants ALORS il y a suffisamment de pharmacies.
S'il y a suffisamment d'écoles, suffisamment de magasins, suffisamment de gendarmeries, suffisamment de casernes de pompiers, suffisamment de centre médicaux et suffisamment de pharmacies ALORS il y a suffisamment de bâtiments publiques.
S'il y a au moins 75% de la superficie de la ville couvert ALORS il y a suffisamment d'habitants.
S'il y a moins de 75% de la superficie de la ville couvert et suffisamment de bâtiments publiques ALORS il faut ajouter des habitants.
S'il y a au moins 75% de la superficie de la ville couvert et suffisamment de bâtiments publiques ALORS la génération de la ville est terminée.

Les différents nombres d'habitants par bâtiments publique sont des estimations de moyenne. Ces derniers peuvent être changer facilement. Si l'on ne remplie la ville qu'a 75% c'est également ici aussi une estimation de la place que l'on doit laisser pour les routes, différents parque et parkings et ainsi que des possibles terrains sans construction.

Si l'on vérifie qu'il y ait suffisamment de bâtiments publics avant d'ajouter d'autres habitants, c'est afin de s'assurer que la superficie de ces derniers soit pris en compte dans la superficie totale des bâtiments de la ville.

II- Présentation des algorithmes

Algorithmes présentés :

- Boucle principale
- Création d'un bâtiment
- Création d'une famille
- Création d'une personne

Boucle principale (DataGeneration.main()) :

La boucle principale du programme va appeler la fonction « diagnose_current_city() » qui va s'occuper de créer les faits initiaux pour l'IA. Ensuite on définit l'ensemble des actions que l'on veut vérifier :

```
fact_to_test = [
    "Add school",
    "Add food shop",
    "Add police station",
    "Add fire station",
    "Add medical office",
    "Add drug store",
    "Add inhabitants",
    "Good city"
]
```

Cette liste représente les conclusions que l'on cherche à faire sur notre ville. L'IA est ensuite appelé avec cette liste de conclusion. Elle retourne une liste composée des bonnes conclusions sur notre ville.

Si notre ville requiert des modifications on appelle alors la fonction « act() » avec en paramètre la liste des conclusions retournée par l'IA. Cette fonction va se charger de créer les bâtiments ou générer de nouveaux habitants selon les besoins. Si aucune modification n'est requise cela signifie que la génération de notre ville est terminée.

Création d'un bâtiment :

Les fonctions assurant la création des bâtiments se trouve dans le module « DBWriter » et s'appellent «create_home() » et «add_public_building() ». La première s'occupe de créer les maisons des habitants, et la seconde de créer les bâtiments d'utilité public.

Un bâtiment possède les propriétés suivantes :

- Un building ID représenté par un entier, ce qui permet à chaque bâtiment de la ville d'avoir un identifiant unique.
- Un numéro de rue représenté par un entier.
- Un numéro d'appartement représenté par un entier, à noter que ce numéro est égal a 0 si le bâtiment en question est d'un type autre que « Flat ».
- Un nom de rue représenté par une chaine de caractères.

- Un code postal représenté par une chaine de caractères, et non par un entier car sous forme d'entier on ne conserve pas les 0 se trouvant devant un nombre.
- Un nombre de pièce représenté par un entier.
- Un type représenté par une chaine de caractére, ce type peut être : House, Flat, School, FoodShop, PoliceStation, FireStation, MedicalOffice et DrugStore
- Un booléen équivalent à vrai si le bâtiment est d'utilité public et a faux dans le cas contraire
- Une taille représentée par un entier, cet entier représente la surface en mètre carré du bâtiment.
- Le nom de la ville à laquelle appartient le bâtiment représenté par une chaine de caractères.

Lors de la création des maisons pour habitants la taille du bâtiment est mise égal a trois fois la taille minimum par habitants afin d'assurer la possibilité au couple d'avoir un enfant.

La taille des bâtiments d'utilité publique est prévue à l'avance dans la fonction « act() » du module « DataGeneration ».

Après la génération d'un bâtiment, ce dernier est ajouté a la base de données.

Création d'une famille :

La création des familles est assurées par la fonction « create_family() » du module « DBWriter ». Afin de garder l'algorithme le plus efficace possible en temps chaque famille est composée de 2 grand pères, 2 grand mères, 1 père, 1 mère et un enfant. Les différente personne sont crée a l'aide de la fonction « create_person() ». Mais il faut s'assurer que chaque membre possède le bon nom de famille ainsi qu'un âge cohérent. C'est pour cela que l'âge de l'enfant est défini au hasard entre 0 et 22 puis on génère ensuite l'âge des parents en prenant l'âge de l'enfant et en ajoutant un nombre aléatoire compris entre 20 et 35. On répète le même procédé pour l'âge des grands-parents cette fois-ci en partant de l'âge du parent correspondant.

Il faut également prendre en compte qu'une même famille doit avoir le même nom de famille.

Une fois les différentes personnes créées dans la base de données, on génère les liens familiaux grâce à la fonction « have_child() » en précisant en paramètre la mère, le père et l'enfant.

Création d'une personne :

La création des personnes est assurée par la fonction « create_person() » du module « DBWriter » elle prend en paramètre l'activité de la personne a créée, son âge puis en paramètre optionnelle : son sexe, son nom de famille, l'id de sa maison. Les paramètres optionnels ont par défaut une valeur de None. Si le sexe n'est pas précisé, il est alors choisi aléatoirement. Si le nom de famille n'est pas précisé, il est alors choisi aléatoirement parmi une liste de nom de famille prédéfini. Et finalement si le bâtiment n'est pas précisé, il est alors crée en utilisant la fonction « create_home() ».

Le prénom est choisi aléatoirement dans une liste de prénom correspondant au sexe de la personne a créée. Son Id est généré avec la fonction « get_possible_id() » du module « DBReader ».

III- Simulations, tests et graphes de déroulement d'une simulation

Pour cet exemple nous allons prendre une ville avec les propriétés suivante :

```
city_information = {
    "CityName":              "Paris",
    "Size":                  20,
    "MinSizePerInhabitant": 10000,
}
```

La surface minimum par habitants est volontairement exagérée afin que l'algorithme de génération ne soit pas trop long.

Une fois la génération terminée on obtient le résultat suivant :

```
Town must have :
        - 1169 inhabitants,
        - 1 food shop,
        - 1 police station,
        - 1 fire station,
        - 1 school,
        - 3 medical office,
        - 2 drug store,
```

Si le programme est lancé dans une console python on peut ensuite vérifier que les prérequis pour chaque bâtiment est complété. Par exemple vérifions si le nombre d'écoles est le bon :

```
In[5]: import DBReader
In[6]: DBReader.get_inhabitants_number_by_activity("Student")
Out[6]: 167
```

Dans les paramètres de l'IA il est défini que chaque école ne peut contenir plus de 500 étudiants, on peut voir ici que le nombre d'étudiant et de 167. La condition est donc validée.

Changeons maintenant la surface minimum par habitant pour obtenir plus d'étudiant.

```
city_information = {
    "CityName":              "Paris",
    "Size":                  20,
    "MinSizePerInhabitant": 500,  # s
}
```

On obtient ainsi :

```
Town must have :
        - 23289 inhabitants,
        - 12 food shop,
        - 2 police station,
        - 2 fire station,
        - 7 school,
        - 47 medical office,
        - 24 drug store,
```

On peut alors vérifier le nombre d'étudiant :

```
In[4]: import DBReader
In[5]: DBReader.get_inhabitants_number_by_activity("Student")
Out[5]: 3327
```

On a maintenant 3327 étudiant ce qui donne : 3327 / 500 = 6 reste 327, il faut donc au moins 7 écoles pour prendre en charge l'ensemble des étudiants.