Compte rendu CSV to graph

Timothé Boyer

October 2023

1 CSV to graphs

Le but de ce sous programme est de transformer un fichier csv en données exploitable pour former un graphe, c'est à dire en liste de noeud de un ensemble de branches que l'on va pondérer avec la durée de chaque tache.

1.1 From CSV

Le but de ce programme est de lire un fichier CSV donner en entrée est de le transformer en liste de Liste ou chaque terme est une liste qui correspond à une ligne et chaque terme de cette liste est un mot de la ligne.
Ce programme nous a été donner dans le sujet et on apporte aucune modification à celui-ci

```
def from_csv(nom_fichier_csv: str)->list:
    """Programme qui ouvre un fichier csv et qui le tranforme en liste de liste mot

    Args:
        nom_fichier_csv (str): Nom du fichier à utiliser

    Returns:
        list: Liste ou chaque terme est une liste qui correspond à une ligne et chaque terme de
                                                        cette liste est un mot de la ligne
    """
    with open(nom_fichier_csv + ".csv" , 'r', encoding="utf8" ) as fichier_csv:
        lecture_fichier_csv =csv.reader(fichier_csv, delimiter= "," ,quoting=csv.QUOTE_ALL)
        l=[]
        for r in lecture_fichier_csv:
            l.append(r)
        return l
```

1.2 Traitement des informations

1.3 Première Version

L'obejectif de ce sous programme est de transformer la liste obtenue en après l'execution de From CSV est de resortir la liste des différents noeuds donc la liste des différentes taches ainsi que les arcs, c'est à dire les branches du graphes donc un ensemble de tupple de 2 éléments qui signifie donc que ces 2 éléments sont reliées. Et enfin la fonction retourne aussi la liste des taches associées à leur durée, c'est à dire une liste de tupples de 2 éléments où le premier est la tache et le second sa durée.

```
    def traitement_information(liste_informations)->tuple[list,set,list]:
    """Fonction qui prend la listes des lignes du fichier csv et qui ressort la liste des
                                                    noeuds du graphe et ses differentes arretes
                                                     ainsi que leur ponderation


    Args:
        liste_informations (list): la liste des lignes d'un fichier

    Returns:
        tuple:
            list: la liste des différents noeuds du graphes
            set: l'ensemble des arcs du graphe
            list: la liste des poids de chaque noeud
                tuple:
                    str: le nom du noeud
                    int: son poids
    """
```

```
    #On creer les différentes variable que l'on retournera
    noeuds=[]
    arcs=set()
    poids=[]
    for ligne in liste_informations: #pour chaque ligne du fichier
        noeud=ligne[0] #le noeud (tache) est la première case de la ligne
        noeuds.append(noeud) #la liste de noeuds prend le noeud
        duree_tache=ligne[2] #la duree de cette tache est la troisième case de la ligne
        for i in range(4,7): #pour chaque colonne de suivi
            if ligne[i]!='': #si il y a un suivi
                duree_tache=ligne[i] #la duree de la tache est la duree dans la dernière
                                                                    colonne de suivi
        duree_tache=conversion_unite(duree_tache) #on convertit la duree qui est une valeur
                                                        suivi d'une unité en jours
        poids.append((noeud,duree_tache))
        if ligne[3]!='': #Si la tache à une tache précédentes
            pre_noeuds=ligne[3] #les pré-taches sont contenus dabs la 4 case de la ligne
            for pre_noeud in pre_noeuds.split(): #pour chaque pré-taches
                arcs.add((pre_noeud, noeud)) #la pré-tache est reliée à la tache
    return noeuds, arcs, poids
```

1.3.1 Conversion des unité

Dans un fichier CSV lorsqu'on indique la durée d'une tache on écrit un nombre suivi d'une unité. Ainsi l'objectif du sous-programme conversion$_u$*niteestderenvoyerlenombredejourscorrespondantladuredonneeentre*.

```
    def conversion_unite(duree_tache):
    """Fonction qui convertie un str d'une duree temporelle et qui le convertir en int qui
                                                    correspond au nombre de jour
    que représente cette duree

    Args:
        duree_tache (str): la duree temporelle avec les unités (mois/annee/semaine/jours)

    Returns:
        int: le nombre de jour qui correspond a la duree
    """
    valeur=float(duree_tache.split()[0])
    unite=duree_tache.split()[1]
    if unite=='mois':
        valeur*=30
    elif unite=='annees' or unite=='annee':
        valeur*=365
    elif unite=='semaines' or unite=='semaine':
        valeur*=7
    elif unite=='jours' or unite=='jour':
        valeur=valeur
    return valeur
```

1.3.2 Seconde Version

Le but de cette version est de pouvoir choisir un suivi des durée. C'est à dire que l'on peut choisir si on prend en compte un jusqu'à un suivi. Cela nous permettra de faire par la suite plusieurs graphes, un pour chaque suivi.

```
    def traitement_information(liste_informations, n_suivi=None)->tuple[list,set,list]:
    """Fonction qui prend la listes des lignes du fichier csv et qui ressort la liste des
                                                    noeuds du graphe et ses differentes arretes
                                                     ainsi que leur ponderation

    Args:
        liste_informations (list): la liste des lignes d'un fichier
        n_suivi(int): le numéro du suivi que l'on veut suivre, si aucune info n_suivi=None =>
                                                        On ne prend pas en compte les suivi

    Returns:
        tuple:
            list: la liste des différents noeuds du graphes
            set: l'ensemble des arcs du graphe
            list: la liste des poids de chaque noeud
                tuple:
                    str: le nom du noeud
```

```
                int: son poids
    """
    #On creer les différentes variable que l'on retournera
    noeuds=[]
    arcs=set()
    poids=[]
    for ligne in liste_informations: #pour chaque ligne du fichier
        noeud=ligne[0] #le noeud (tache) est la première case de la ligne
        noeuds.append(noeud) #la liste de noeuds prend le noeud
        duree_tache=ligne[2] #la duree de cette tache est la troisième case de la ligne
        if n_suivi!=None: #Si on s'interesse au suivi
            for i in range(4,5+n_suivi): #pour chaque colonne de suivi
                if ligne[i]!='': #si il y a un suivi
                    duree_tache=ligne[i] #la duree de la tache est la duree dans la dernière
                                                                    colonne de suivi
        duree_tache=conversion_unite(duree_tache) #on convertit la duree qui est une valeur
                                                    suivi d'une unité en jours
        poids.append((noeud,duree_tache))
        if ligne[3]!='': #Si la tache à une tache précédentes
            pre_noeuds=ligne[3] #les pré-taches sont contenus dabs la 4 case de la ligne
            for pre_noeud in pre_noeuds.split(): #pour chaque pré-taches
                arcs.add((pre_noeud, noeud)) #la pré-tache est reliée à la tache
    return noeuds, arcs, poids
```

1.4 Pondération des branches