Université de Bordeaux Année 2018 - 2019
Master 2 Bio-Informatique

Visualisation d'Information
Projet – Analyse visuelle de données d'expression génique –

Objectif : L'objectif de ce projet est de réaliser une analyse d'un jeu de données qui vous est fournit. Pour cela, vous trouverez ci-dessous plusieurs sous-objectifs à atteindre (le barème est donné à titre indicatifs). Le résultat sera remis sous la forme d'un fichier contenant le code python (compatible avec l'IDE Python de Tulip) permettant à partir d'un fichier au format Tulip (cf ci-après) d'atteindre ces différents sous-objectifs. Vous devrez aussi fournir un rapport d'une dizaine de pages expliquant les algorithmes, choix que vous aurez faits et informations tirées de l'analyse que vous ferez des données.

Modalités : Projet à réaliser en binôme

Dates importantes : remise du code 3 février à 23h59 et soutenance le 4 février

Fichiers fournis : Vous trouverez à l'adresse :
www.labri.fr/perso/bourqui/downloads/cours/Master/2018/Projet/
un fichier contenant un réseau de régulation de gènes d'Escherichia coli K12 qui vous servira dans ce projet. Comme vous pourrez le remarquer, ce graphe possède deux sous-graphes, *clone* et *Genes interactions*. Le premier est une copie du jeu de données initial, a priori il ne faudra pas modifier ce graphe. Le deuxième possède un certain nombre de sous-graphes obtenus en le partitionnant de manière récursive.

Partie 1 : Pré-traitement & première visualisation (/2 points)

Comme vous pouvez le voir après le chargement du réseau fourni, le graphe tel qu'il est affiché n'apporte que peu d'information. Écrire le code python permettant d'affecter :

- des étiquettes aux sommets du réseaux (les locus seront utilisés)
- une taille (non-nulle) à chaque sommet afin de pouvoir visualiser correctement les étiquettes
- des couleurs/formes différentes pour les différents types de régulation (les arêtes de ce réseau)
- des positions aux sommets du graphe (le choix et le paramétrage de l'algorithme de dessin est libre)

Partie 2 : Dessin *faisceauté* du réseau d'interactions (/8 points)

Question 2.1 : Écrire un algorithme permettant de construire l'arbre hiérarchique représentant le partitionnement du réseau d'interactions. Cet arbre devra être un sous-graphe du graphe racine, ses sommets internes représenterons les différents *clusters* (sous-graphes) et ses feuilles seront exactement l'ensemble des sommets du réseau d'interactions. Pour tout sommet u et v de cet arbre, il existe un arc (u,v) si et seulement si le cluster représenté par u contient *directement* le cluster (ou le sommet du réseau d'interactions) représenté par v.

Question 2.2 : Ecrire un algorithme (fonction) permettant de dessiner de manière radiale l'arbre hiérarchique (pensez à utiliser les plugins de Tulip).

NB : vous pourrez remarquer que dessiner l'arbre dessine aussi le réseau d'interactions car ces deux graphes partagent la même propriété de positionnement (LayoutProperty).

Question 2.3 : Ecrire un algorithme qui étant donnés un graphe et une propriété (de type DoubleProperty) permet de colorer les sommets de ce graphe en fonction de leur valeur pour cette propriété.

Question 2.4 : Ecrire un algorithme qui permet de construire des faisceaux d'arêtes (*bundles*). L'idée de l'algorithme est de router chaque arête du réseau d'interactions en utilisant la topologie et le dessin de l'arbre. Ainsi pour une arête (u,v), il faut :

1. Calculer dans l'arbre le chemin reliant u à v
2. Utiliser les positions des sommets de cette séquence comme points de contrôle pour l'arêtes

Pour cela, écrire :

a) Un algorithme qui, étant donnés un arbre et deux sommets, calcule et retourne la séquence de sommets du plus court chemin de u à v
b) Un algorithme qui construit les faisceaux d'arêtes en affectant à chaque arêtes ses points de contrôle

Partie 3 : Construction d'imagettes (/6 points)

Une technique connue pour permettre de visualiser correctement les données dynamiques sont les *small multiples* (imagettes en français). Dans cette technique, une imagette est construite pour chaque pas de temps des données puis toutes les imagettes sont positionnées dans une grille.

Question 3.1 : Écrire un algorithme qui produit construit un graphe et une hiérarchie de sous-graphes définis comme suit :

1) Le graphe « smallMultiples » devra être un sous-graphe du graphe racine
2) Pour chaque pas de temps (stocké dans les propriétés tp_*, cf ci-dessous), construit un sous-graphe *copie* du graphe d'interactions (copie des propriétés de taille, couleur, position, étiquette, forme) et affecte à la propriété viewMetric de ce graphe les valeurs d'expression du pas de temps courant

NB : vous pourrez remarquer que chaque imagette est alors correctement dessinée mais elles ne sont pas encore positionnées les unes par rapport aux autres

Question 3.2 : Écrire un algorithme permettant de colorer chacune des imagettes en fonction de l'expression de gènes.

Question 3.3 : Ecrire un algorithme permettant de positionner les imagettes dans une grille dont le

nombre de colonnes sera un paramètre (attention à respecter l'ordre des pas de temps).

Question 3.4 : Ecrire un algorithme qui construit les imagettes.

Partie 4 : Analyse des données (/4 points)

Cette partie est laissée relativement libre, l'objectif est de permettre de comprendre l'évolution des niveaux d'expression de gène et dans quels processus biologiques ces gènes sont impliqués.

Vous pouvez utiliser n'importe quelle source de données externe dans la mesure où cela est fait de manière programmatique (vous pourrez aussi utiliser vos propres connaissances). Vous pourrez aussi programmer n'importe quelle autre représentation, partitionnement (le partitionnement fournit permet-il une interprétation ?), extraction de sous-parties sur/sous exprimées, *etc*...

Les résultats de cette partie devront être décrits dans le rapport.

Partie 5 : Pour aller plus loin [Bonus]

Question 5.1 : L'algorithme de dessin radial fourni par Tulip ne positionne pas les feuilles de l'arbre sur un cercle. Proposer un algorithme de dessin respectant cette contrainte.

Question 5.2 : Le partitionnement fourni ne permet peut-être pas de répondre aux questions que l'on se pose. Ecrire un algorithme permettant de partitionner de manière hiérarchique le graphe d'interactions (pensez à utiliser les algorithmes de partitionnement fournis par Tulip).

Description du format de graphe :

Dans le fichier fourni, un certain nombre de propriétés existent (en plus des propriétés visuelles, commençant par « view* ») :

Nom	Type	Description
Locus	StringProperty	Spécifie pour chaque sommet un locus.
tp* s	DoubleProperty	Spécifie pour chaque sommet le niveau d'expression du gène correspondant à ce « time point ».
Positive	BooleanProperty	Spécifie pour chaque arête si elle représente une régulation positive.
Negative	BooleanProperty	Spécifie pour chaque arête si elle représente une régulation négative.

Voici un extrait vous donnant quelques indications sur le jeu de données utilisé [WvHK08] :
« The dataset contains gene expression data from 17 time points, during which E. coli is grown on a mixture of glucose and lactose. The bacterium grows preferentially on glucose until that energy source

is depleted, resulting in growth arrest while the cells adjust to growth on lactose. This shift, called the diauxic shift, takes places at about time point 6. At time point 14, the stationary phase is entered in which the organism stops growing due to the lack of nutrients. During this phase, many processes are shut down by the bacterial cell in order to save energy. »

[WvHK08] M. A. Westenberg, S. A. F. T. van Hijum, O. P. Kuipers, J. B. T. M. Roerdink. Visualizing Genome Expression and Regulatory Network Dynamics in Genomic and Metabolic Context. *Computer Graphics Forum*, **27**(3):887-894, 2008.