Implémentation du modèle d'Hegselmann-Krause avec MPI.

Professeur Frank Nielsen

Achari Berrada Youssef, X2013.

INF442

Présentation du modèle :

Le modèle de Hegselmann-Krause ou HK décrit la dynamique dans un système d'influence à n agents $X=\{x_1,..,x_n\}$. Chaque agent x_i est caractérisé par d attributs réels $\{x_i^1,..,x_i^d\}$. À partir d'un instant initiale t=0, jusqu'à la convergence du modèle, un état du système est définit par l'ensemble des attributs des n agents. On note alors $S(t)=\{x_1(t),.., x_2(t)\}$. L'évolution d'un tel système en fonction du temps est décrit par le modèle d'influence HK suivant :

1. À chaque pas de temps, un agent adopte un nouvel état qui est la moyenne arithmétique des états des agents qui lui sont voisins et de son état lui même. Par la suite nous allons considéré que deux agents sont voisins si est seulement si la distance entre ses deux agents est inférieure à un seuil donné $\epsilon > 0$. Nous remarquons que c'est une relation symétrique et réflexive mais pas transitive. Pour cela nous allons construire la fonction voisin qui prend en argument un seuil et qui pour chaque agent associe ses voisins. Il est important de remarquer que un agent est toujours voisin de lui même (refléxivité).
   On notera : $G_i(t)$ les voisins de l'agent i à l'instant t, et $n_i(t) = |G_i(t)|$ le nombre de voisin de l'agent i.
   Ainsi nous obtenons la formule suivante :
   $$x_i(t+1) = \frac{\sum_{j \in G_i(t)} x_i}{n_i(t)}$$
2. Au bout d'un certain temps, le modèle converge lorsque l'état du système n'évolue plus en fonction du temps. Comme le montre [1], la convergence peut être montrer de plusieurs façons notamment en utilisant les propriétés des matrices stochastiques ou alors d'exhiber une fonction de Lyapounov qui décroit en chaque étape.

Classes utilisés :

Pour implémenter notre modèle d'influence HK sur un système d'agent, nous allons considéré un ensemble de classe en C++ pour représenter l'évolution du système.

Agent :

Private : double* attributs, int d # nombre de réels caractérisant un agent.
Public : Agent(double* attributs, int d) # constructeur d'un agent.
double distanceE(& Agent)# Distance euclidienne entre deux données.

Etat :

Private : vector<Agent*> agents, int n # nombre d'agents dans notre système.
Public : vector<vector<int>> voisin(double seuil)
Etat etatSuivant();

HKAlgo : Classe contenant le main.

Parallélisation avec MPI :

Une propriété importante va nous permettre de paralléliser notre fonction de communication est la décomposabilité de la fonction voisin sur une partition de X. En effet, soit $X = X_1 \uplus X_2$, alors on a

$$voisin_{seuil}(x, X) = voisin_{seuil}(x, X_1) \cup voisin_{seuil}(x, X_2)$$

Veuillez noter que x est forcément présent soit dans X_1 soit dans X_2 mais pas les deux.
Cette relation est similaire mais pas toute à fait la même que celle rencontrer avec les k-plus proche voisin, où on avait :

$$NN_k(x, X) = NN_k(x, NN_k(x, X_1) \cup NN_k(x, X_2))$$

car ici il n'y a pas de limitation sur le nombre de voisins que peut avoir x si ce n'est que par le seuil.
Ainsi l'algorithme parallèle peut être construit de la façon suivante :

- Pour p processeurs, on partitionne donc X en paquets de X_i de taille $\frac{n}{p}$, et on fait dans chaque processus en parallèle la requête $voisin_{seuil}(x, X_i)$ dans chaque paquet en parallèle localement.
- Ensuite le processus maître reçoit les éléments pour les unifier dans une variable globale.
- Cacluler le nouvel état qui peut aussi être fait en parallèle.
- Refaire une boucle jusqu'à la convergence et la stabilisation de l'évolution des états.

Comparaison de la compléxité entre l'algorithme séquentiel et l'algorithme parallèle :

- Séquentiel : $\frac{O(d\,n^2)}{recherche\ voisins} + \frac{O(d\,n^2)}{calcul\ \ état\ suivant} = O(d\,n^2)$
- Parallèle : $\frac{O(\frac{d\,n^2}{p})}{recherche\ voisins} + \frac{O(\frac{d\,n^2}{p})}{calcul\ \ état\ suivant} = O(\frac{d\,n^2}{p})$

Analogie avec kNN :

kNN	HK
- Décomposable donc parallélisable. - Label. - On cherche à prédire le label d'une nouvelle donnée à partir d'un *Training Set* à n données. -Erreur de précision sur la nouvelle donnée. -Utiliser dans la classification, Bonne heuristique pour le partitionnement de Voronoi.	-Décomposable donc parallélisable. -Pas de label -On cherche à estimer l'évolution d'un système d'influence de n agents. -Erreur d'estimation sur les n agents mais qui nécessite la connaissance de l'état du futur. -Utiliser en politique, physique pour étudier les systèmes d'influence. -Un agent est auto-influençable.

Bibliographie :

[1] P. Dejean De La Bâtie, T. Grange, J. Pan, S. Petroff, V. Senlis, Rapport de PSCX2013 : Étude et modélisation des systèmes d'influences à partir du modèle d'Hegselmann-Krause, 2014-2015.

[2] Hegselmann, R. and Krause, U. [2002] “Opinion Dynamics and Bounded Confidence: Models, Analysis, and Simulations,” Journal of Artificial Societies and Social Simulation 5, available at http://jasss.soc.surrey.ac.uk/5/3/2.html.