Projet de Master — Simulation de recherche publicitaire sur graphe pondéré

Contexte général

Dans les systèmes de recommandation ou de ciblage publicitaire, chaque utilisateur, produit ou contenu peut être représenté par un **ensemble de caractéristiques numériques** (intérêts, âge, affinité, style de navigation, budget, etc.).

On cherche à simuler ce problème sous forme d'un **graphe logique**, où les **nœuds** représentent des entités (utilisateurs, produits, annonces, etc.) et les **arêtes** traduisent une relation de similarité, de proximité ou de compatibilité.

Le problème fondamental à résoudre est de déterminer **quels nœuds du graphe** se trouvent dans une **zone d'intérêt** autour d'un nœud donné, selon un **vecteur de pondération** qui reflète l'importance relative de chaque caractéristique.

Objectif du projet

Étant donné :

- un graphe $G = (V, E)$,
- un nœud de départ $A \in V$,
- un vecteur de pondération $Y = (y_1, y_2, \ldots, y_{50})$,
- un rayon $X > 0$,

on veut trouver tous les nœuds $v \in V$tels que :

$$d_Y(A, v) = \sqrt{\sum_{k=1}^{50} y_k (f_{Ak} - f_{vk})^2} \leq X$$

où $F(v) = (f_{v1}, \ldots, f_{v50})$est le vecteur de caractéristiques du nœud v.

Ce problème correspond à une **recherche pondérée dans un espace de grande dimension**, combinée à une **structure de graphe librement définie**.

Travail demandé

Étape 1 — Construction du graphe

- Vous devez **concevoir un graphe G** contenant entre **500 et 5000 nœuds**.

- Chaque nœud v_i est représenté par un vecteur de 50 caractéristiques numériques réelles, générées aléatoirement ou selon une logique simulant une base publicitaire (ex. : affinité, âge, revenu, thématique, etc.).
- Vous pouvez définir les arêtes E selon :
  - une proximité logique (distance entre vecteurs),
  - ou une structure arbitraire (connexions aléatoires, hiérarchiques, sectorielles, etc.).

Étape 2 — Implémentation de la distance pondérée

- Implémentez la fonction de distance :

$$d_Y(u, v) = \sqrt{\sum_{k=1}^{50} y_k (f_{uk} - f_{vk})^2}$$

- Vérifiez que votre implémentation supporte les variations de pondération Y et permet des recherches efficaces.

Étape 3 — Recherche dans le rayon X

- Étant donné un nœud de départ A, un vecteur Y et un rayon X, trouvez **tous les nœuds du graphe situés à distance pondérée ≤ X**.
- Vous pouvez comparer différentes stratégies :
  - **naïve** : parcours exhaustif de tous les nœuds,
  - **structurée** : parcours restreint par arêtes, KD-tree, graph partitioning, etc.

Étape 4 — Heuristique d'optimisation

- Le problème devient **combinatoire** pour de grands graphes et des vecteurs à 50 dimensions.
- Proposez **une amélioration algorithmique** : par exemple :
  - réduction de dimension (PCA),
  - clustering préalable,
  - recherche approximative (ANN),
  - filtrage adaptatif selon les coefficients de Y.

Étape 5 — Extension (optionnelle, bonus)

- Étudiez le problème suivant :

  Trouver le **chemin de similarité maximale** reliant A à un nœud cible B, où chaque arête a un coût défini par la distance pondérée.

- Montrez que ce problème est **NP-difficile**, et proposez une **heuristique ou approximation**.

Résultats attendus

Vous devez produire :

1. Un rapport de **10 à 15 pages** contenant :
    - description du modèle de graphe choisi ;
    - définitions mathématiques utilisées ;
    - complexité de l'algorithme développé ;
    - tests expérimentaux (temps de calcul, performance, précision) ;
    - discussion sur les limites du modèle.
2. Un code fonctionnel (Python recommandé, mais libre) avec un script de démonstration.
3. Un court résumé (1 page) expliquant en quoi votre approche pourrait s'appliquer à la recherche publicitaire réelle.

Critères d'évaluation

Critère	Pondération
Qualité de la modélisation du graphe	20 %
Correctitude de la distance pondérée	15 %
Performance et justesse de l'algorithme de recherche	25 %
Innovation ou heuristique d'optimisation	20 %
Qualité du rapport	20 %

Variante 1 — Orientée Algorithmique (informatique / IA)

Objectif : performance, complexité, heuristiques de recherche.

- Implémenter et comparer plusieurs stratégies : BFS pondéré, A*, Dijkstra, KD-tree, etc.
- Étudier la complexité moyenne et asymptotique de vos approches.
- Justifier pourquoi la recherche exacte est NP-difficile.

Livrables finaux

- **Code source** commenté (.py, .sur github)
- **Rapport PDF**
- **Résumé synthétique**
- (Optionnel) Présentation orale ou démo sur teams (10 min)