Master Informatique

Projet d'Algorithmique et complexité

Année Académique 2024-2025

Optimisation de Réseaux Logistiques : Algorithmes et Structures de Données pour Systèmes de Distribution Complexes

Responsable du module : Dr. ATTA Amanvon Ferdinand

Date de remise : **10 Avril 2025 à 23h59**

Mode de travail : Groupes de 10 étudiants

1 Introduction

Ce document présente le projet principal du module d'Algorithmique et complexité du Master Informatique (CIO,BDA, D2A, BC) à l'Université Virtuelle de Côte d'Ivoire pour l'année académique 2024-2025. Ce projet vise à consolider les connaissances théoriques acquises en cours et à développer des compétences pratiques dans l'implémentation et l'analyse d'algorithmes complexes.

2 Objectifs pédagogiques

À l'issue de ce projet, les étudiants devront avoir acquis :

- Une maîtrise approfondie des listes chaînées et de leur implémentation en langage C
- Une compréhension pratique des algorithmes de parcours de graphes (DFS, BFS)
- Des compétences solides en programmation dynamique
- La capacité à mettre en œuvre des approches gloutonnes pour résoudre des problèmes d'optimisation
- Une compréhension des principes et de l'implémentation des algorithmes génétiques
- L'aptitude à analyser la complexité algorithmique et à évaluer empiriquement les performances

3 Contexte du projet

3.1 Problématique

Dans un contexte d'expansion économique en Côte d'Ivoire et dans la sous-région ouest-africaine, les entreprises de transport et de logistique font face à des défis croissants d'optimisation. La complexité des réseaux de distribution, les contraintes d'infrastructure, la variabilité des conditions de transport et les exigences de service client nécessitent des solutions algorithmiques avancées.

Ce projet s'inspire d'un cas concret d'une entreprise logistique opérant en Afrique de l'Ouest qui doit optimiser son réseau de distribution couvrant zones urbaines denses (Abidjan, Yamoussoukro), zones rurales et connexions internationales.

3.2 Description du réseau logistique

Le réseau logistique à modéliser comprend :

- **Centres de distribution principaux** : Grands hubs situés dans les principales villes (capacité et services étendus)
- **Points de relais intermédiaires** : Centres de taille moyenne permettant le tri et la redistribution
- **Points de livraison finaux** : Clients particuliers et professionnels
- **Routes avec attributs multiples** :
  - Distance kilométrique

— Temps de parcours estimé (variable selon l'heure)
— État de la route (asphalté, latérite, piste)
— Fiabilité saisonnière (certaines routes impraticables en saison des pluies)
— Coûts (carburant, péages, maintenance)
— Restrictions (poids maximum, hauteur, largeur)

3.3 Contraintes spécifiques

Le réseau présente plusieurs particularités à prendre en compte :

— **Structure dynamique** : Certaines routes peuvent être temporairement inaccessibles (travaux, événements, inondations)
— **Contraintes temporelles** : Fenêtres de temps pour les livraisons, heures d'ouverture des points de relais
— **Flotte hétérogène** : Véhicules de différentes capacités, consommations et aptitudes (tout-terrain, réfrigéré)
— **Priorités diverses** : Colis standard, express, périssables, fragiles
— **Saisonnalités** : Variations importantes de volume pendant certaines périodes (fêtes, rentrées scolaires)

Ce réseau peut être modélisé comme un graphe orienté pondéré multi-attributs, où les poids des arêtes représentent différents types de coûts pouvant évoluer dans le temps.

4 Travail demandé

4.1 Modélisation du réseau sous forme de graphe

4.1.1 Représentation en liste chaînée d'adjacence

— Implémenter une structure de nœud avec des listes chaînées pour représenter les connexions entre les différents points du réseau
— Concevoir une structure capable de stocker des arêtes avec attributs multiples (distance, temps, coût, restrictions, fiabilité)
— Développer les fonctions de base pour la manipulation du graphe (création, ajout/-suppression de nœuds et d'arêtes)
— Implémenter un mécanisme permettant de modéliser les variations temporelles des attributs (congestion aux heures de pointe, indisponibilité saisonnière)

```
// Structure pour les attributs d'une arete
typedef struct EdgeAttr {
    float distance;      // en kilometres
    float baseTime;      // en minutes (temps nominal)
    float cost;          // cout monetaire
    int roadType;        // type de route (0: asphalte, 1: laterite
       , etc.)
    float reliability;   // indice de fiabilite [0,1]
    int restrictions;    // restrictions codees en bits
    // Autres attributs pertinents
```

```c
} EdgeAttr;

// Structure pour un noeud de la liste d'adjacence
typedef struct AdjListNode {
    int dest;            // identifiant du noeud destination
    EdgeAttr attr;       // attributs de l'arete
    struct AdjListNode* next; // pointeur vers le prochain noeud
} AdjListNode;

// Structure pour la liste d'adjacence
typedef struct AdjList {
    AdjListNode* head;   // tete de la liste
} AdjList;

// Structure pour le graphe
typedef struct Graph {
    int V;               // nombre de sommets
    AdjList* array;      // tableau des listes d'adjacence
    // Informations supplementaires sur les sommets
} Graph;
```

Listing 1 – Structure suggérée pour la représentation du graphe

4.1.2 Gestion des données du réseau

- Développer un parseur pour charger le graphe à partir de fichiers au format JSON ou XML
- Implémenter des fonctions pour sauvegarder l'état du réseau
- Créer un module simple de visualisation textuelle du réseau (représentation par matrice ou par listes)

4.2 Implémentation des algorithmes de parcours de graphe

4.2.1 Algorithmes de base

- Implémenter l'algorithme de parcours en profondeur (DFS) vu au cours
- Implémenter l'algorithme de parcours en largeur (BFS) vu au cours
- Documenter précisément les structures auxiliaires utilisées (pile, file) et justifier les choix d'implémentation

4.2.2 Applications des parcours

- Utiliser les algorithmes de parcours pour :
  - Détecter les cycles dans le réseau
  - Identifier les composantes connexes et les points d'articulation
  - Déterminer l'accessibilité entre différentes zones du réseau
  - Calculer des statistiques sur la connectivité du réseau

4.3 Optimisation des trajets

4.3.1 Approche par programmation dynamique

- Implémenter l'algorithme de Floyd-Warshall pour calculer les plus courts chemins entre toutes les paires de sommets
- Adapter l'algorithme de Bellman-Ford pour gérer les contraintes temporelles et les coûts variables
- Concevoir une solution au problème du voyageur de commerce pour optimiser les tournées de livraison
- Développer un algorithme de programmation dynamique pour la planification multi-jours des livraisons

4.3.2 Approche gloutonne

- Concevoir et implémenter un algorithme glouton pour :
  - L'affectation des colis aux véhicules
  - La planification des tournées dans une journée
  - La redistribution dynamique en cas d'imprévu

4.3.3 Algorithme génétique

- Concevoir une représentation chromosomique adaptée au problème d'optimisation de tournées de véhicules
- Développer les opérateurs génétiques spécifiques :
  - Croisement adapté aux contraintes du problème
  - Mutation intelligente préservant la validité des solutions
  - Fonction de fitness multi-critères
- Implémenter un mécanisme d'élitisme et de sélection par tournoi
- Concevoir des stratégies d'adaptation des paramètres génétiques au cours de l'évolution

4.4 Expérimentation et analyse

4.4.1 Jeux de données et scénarios

- Créer au moins trois jeux de données représentatifs :
  - Un petit réseau (10-20 nœuds) pour les tests initiaux
  - Un réseau moyen (100-150 nœuds) pour les évaluations intermédiaires
  - Un grand réseau (200+ nœuds) pour tester le passage à l'échelle
- Concevoir des scénarios réalistes :
  - Jour normal d'activité
  - Période de pointe (fêtes, événements)
  - Situation de crise (routes coupées, indisponibilités)

4.4.2 Évaluation des performances

- Mesurer et comparer systématiquement :
  - Temps d'exécution des différents algorithmes
  - Qualité des solutions obtenues (distance, temps, coût)
  - Utilisation mémoire
  - Stabilité face aux variations des données d'entrée
- Analyser l'impact de la taille et de la structure du réseau sur les performances
- Étudier les compromis entre rapidité d'exécution et qualité des solutions

4.4.3 Analyse de complexité

- Pour chaque algorithme implémenté :
  - Justifier théoriquement la complexité temporelle dans le pire cas

5 Organisation du travail

5.1 Livrables attendus

1. **Code source** complet, documenté et modulaire en langage C
2. **Rapport technique** (30-40 pages) comprenant :
   - Introduction et analyse du problème
   - Description des structures de données utilisées
   - Présentation des algorithmes implémentés
   - Analyse des résultats expérimentaux
   - Étude comparative des différentes approches
   - Conclusion et perspectives d'amélioration
3. **Manuel d'utilisation** expliquant l'installation et l'exécution du programme
4. **Jeux de données** utilisés pour les tests
5. **Présentation** pour la soutenance finale (20 minutes + 10 minutes de questions)

6 Critères d'évaluation

L'évaluation portera sur les aspects suivants :

- **Qualité du code** (20%) :
  - Correction des implémentations
  - Modularité et réutilisabilité
  - Documentation et lisibilité
  - Gestion des erreurs
- **Rapport technique** (20%) :
  - Clarté et précision des explications

- Pertinence de l'analyse algorithmique
- Qualité de l'analyse des résultats
- Rigueur scientifique

— **Fonctionnalités et performances** (20%) :

- Conformité aux spécifications
- Efficacité des algorithmes
- Qualité des solutions obtenues
- Passage à l'échelle

— **Soutenance** (40%) :

- Clarté de la présentation
- Qualité des réponses aux questions
- Équilibre de participation des membres
- Respect du temps imparti

7 Annexes

7.1 Format des fichiers d'entrée

Un exemple du format JSON attendu pour décrire le réseau :

```
{
  "nodes": [
    {
      "id": 0,
      "name": "Hub Abidjan",
      "type": "hub",
      "coordinates": [5.359952, -4.008256],
      "capacity": 1000
    },
    {
      "id": 1,
      "name": "Relais Yamoussoukro",
      "type": "relay",
      "coordinates": [6.827623, -5.289343],
      "capacity": 500
    },
    // ...autres n uds
  ],
  "edges": [
    {
      "source": 0,
      "destination": 1,
      "distance": 248.5,
      "baseTime": 180,
      "cost": 25000,
      "roadType": 0,
      "reliability": 0.95,
```

```
      "restrictions": 3,
      "timeVariation": {
        "morning": 1.2,
        "afternoon": 1.5,
        "night": 0.9
      }
    },
    // ...autres aretes
  ]
}
```

Listing 2 – Exemple de format JSON pour le réseau

7.2 Exemple de scénario de test

```
SCENARIO: jour_normal
DATE: 2025-02-15
CONDITIONS:
- Toutes les routes disponibles
- Trafic normal

DEMANDES:
- Origine: 0, Destination: 5, Volume: 120, Priorite: standard, Deadline: 18:00
- Origine: 1, Destination: 3, Volume: 80, Priorite: express, Deadline: 14:00
- Origine: 2, Destination: 7, Volume: 60, Priorite: fragile, Deadline: 16:00
// ...autres demandes

VEHICULES:
- Type: camion, Capacite: 300, Disponibilite: 08:00-18:00, Cout: 500/km
- Type: utilitaire, Capacite: 100, Disponibilite: 07:00-20:00, Cout: 300/km
// ...autres vehicules
```

Listing 3 – Exemple de format pour un scénario de test