Institut National des Sciences Appliquées
et de Technologie
INSAT

Projets de Recherche Opérationnelle

5 Projets d'Optimisation avec Gurobi

Réalisé par :

Elyes Mlawah	Gestion de Flotte
Makki Aloulou	Chemin Optimal avec Checkpoint
Mohamed Yassine Kallel	Optimisation de Réseau de Transport
Aymen Abid	Planification Financière et Opérationnelle
Ahmed Loubiri	Ordonnancement de Camions sur Quais

Classe :

GL3

Année Universitaire 2025-2026

Membres du Groupe

Elyes Mlawah

Projet 1 : Gestion de Flotte

Makki Aloulou

Projet 2 : Chemin avec Checkpoint

Mohamed Yassine Kallel

Projet 3 : Réseau de Transport

Aymen Abid

Projet 4 : Planification Financière

Ahmed Loubiri

Projet 5 : Ordonnancement de Camions

Table des matières

1 Introduction Générale

1.1 Contexte

La **Recherche Opérationnelle** est une discipline mathématique qui vise à optimiser des processus complexes en utilisant des modèles mathématiques et des algorithmes performants. Dans le cadre de notre formation en GL3, nous avons développé **5 projets complémentaires** illustrant différentes applications de la **Programmation Linéaire en Nombres Entiers (PLNE)**.

1.2 Présentation des 5 Projets

Ce rapport présente cinq projets distincts, chacun résolvant un problème d'optimisation spécifique :

1. **Projet 1 - Gestion de Flotte (Elyes)**
   - **Problème :** Optimisation de l'affectation de commandes à des camions et chauffeurs
   - **Type :** Vehicle Routing Problem (VRP) avec tournées multi-commandes
   - **Variables :** 225 variables binaires (affectation + utilisation)
2. **Projet 2 - Chemin Optimal avec Checkpoint (Makki)**
   - **Problème :** Trouver le chemin le moins cher passant par au moins un checkpoint
   - **Type :** Plus court chemin avec contraintes de passage obligatoire
   - **Variables :** Variables binaires pour arêtes et checkpoints
3. **Projet 3 - Réseau de Transport (Yassine)**
   - **Problème :** Conception optimale d'un réseau de transport
   - **Type :** Network Design Problem avec nœuds d'offre et de demande
   - **Variables :** Routes, capacités et coûts de transport
4. **Projet 4 - Planification Financière (Aymen)**
   - **Problème :** Optimisation de la production, stocks et investissements
   - **Type :** Planification multi-périodes avec décisions d'investissement
   - **Variables :** Production, approvisionnement, stocks, capacité
5. **Projet 5 - Ordonnancement de Camions (Ahmed)**
   - **Problème :** Ordonnancement de camions sur quais de chargement
   - **Type :** Ordonnancement sur machines parallèles avec contraintes
   - **Variables :** Affectation, séquencement, temps de début

1.3 Objectifs Communs

Tous les projets partagent les objectifs suivants :

— **Modélisation mathématique rigoureuse** en PLNE

— **Résolution optimale** avec le solveur Gurobi

— **Interface graphique** moderne avec PyQt5

— **Visualisation** des résultats et des solutions

1.4 Structure du Rapport

Ce rapport est organisé en 4 parties principales, une pour chaque projet, suivies d'une conclusion comparative.

2 Projet 1 : Gestion de Flotte

2.1 Contexte du Projet

La gestion optimale d'une flotte de transport est un problème crucial dans le domaine de la logistique. Les entreprises de transport doivent quotidiennement affecter des commandes à des véhicules et des chauffeurs tout en minimisant les coûts et en respectant de nombreuses contraintes opérationnelles.

2.2 Problématique

Soit une entreprise de transport disposant de :

— $\mathbf{n_t}$ camions de différents types (Standard, Réfrigéré, Citerne, etc.)

— $\mathbf{n_d}$ chauffeurs avec des permis et tarifs variés

— $\mathbf{n_o}$ commandes à livrer avec des caractéristiques spécifiques

Objectif : Affecter les commandes aux couples (camion, chauffeur) de manière à minimiser le coût total tout en respectant toutes les contraintes.

2.3 Approche Utilisée

Nous avons modélisé ce problème comme un **Programme Linéaire en Nombres Entiers (PLNE)** avec variables binaires, résolu par l'algorithme **Branch & Bound** implémenté dans Gurobi.

2.4 Spécificité : Tournées Multi-Commandes

Innovation du Modèle

Contrairement aux modèles classiques d'affectation simple (1 camion = 1 commande), notre système permet à **un même camion de transporter plusieurs commandes simultanément** lors d'une même tournée, ce qui optimise l'utilisation des ressources et réduit les coûts.

Principe :

— Un camion peut prendre **jusqu'à** max_orders **commandes** par tournée

— Les commandes sont regroupées si elles respectent les contraintes de capacité

— Le coût est calculé pour l'ensemble de la tournée

Exemple de Tournée Multi-Commandes

Route 3 : Camion C003 (Citerne) avec Chauffeur D002

- **Commande O004 :** 3 tonnes, 150 km
- **Commande O005 :** 2.5 tonnes, 120 km
- **Commande O006 :** 1.8 tonnes, 90 km

Vérifications :

- Poids total : $3 + 2.5 + 1.8 = 7.3$ tonnes ≤ 12 tonnes ✓ OK
- Nombre commandes : $3 \leq 5$ (max_orders) ✓ OK
- Type compatible : Toutes sont "Liquide" ✓ OK

Coût total : 420.3 TND (au lieu de 3 trajets séparés)

Avantages :

1. **Réduction des coûts** : Moins de trajets = moins de carburant
2. **Optimisation des ressources** : Meilleure utilisation de la capacité des camions
3. **Efficacité logistique** : Moins de véhicules mobilisés
4. **Impact environnemental** : Réduction des émissions CO_2

2.5 Formulation Mathématique

2.5.1 Ensembles et Indices

Ensembles

$$\begin{aligned} T &= \{1, 2, \ldots, n_t\} \quad \text{Ensemble des camions} \\ D &= \{1, 2, \ldots, n_d\} \quad \text{Ensemble des chauffeurs} \\ O &= \{1, 2, \ldots, n_o\} \quad \text{Ensemble des commandes} \end{aligned}$$

2.5.2 Paramètres

2.5.3 Variables de Décision

Variables Binaires

$$x_{t,d,o} \in \{0, 1\} \quad \forall t \in T, d \in D, o \in O$$

$$\text{où } x_{t,d,o} = \begin{cases} 1 & \text{si camion } t \text{ avec chauffeur } d \text{ transporte commande } o \\ 0 & \text{sinon} \end{cases}$$

$$y_{t,d} \in \{0, 1\} \quad \forall t \in T, d \in D$$

$$\text{où } y_{t,d} = \begin{cases} 1 & \text{si camion } t \text{ est utilisé avec chauffeur } d \\ 0 & \text{sinon} \end{cases}$$

Paramètre	Description
$capacity_t$	Capacité du camion t (tonnes)
$cost\_per\_km_t$	Coût par kilomètre du camion t (TND/km)
max_orders_t	Nombre max de commandes pour le camion t
$compatible\_types_t$	Types de marchandises compatibles avec t
$hourly\_rate_d$	Tarif horaire du chauffeur d (TND/h)
$available_d$	Disponibilité du chauffeur d (booléen)
$can\_drive_{d,t}$	Chauffeur d peut conduire camion t
$weight_o$	Poids de la commande o (tonnes)
$distance_o$	Distance de la commande o (km)
$order\_type_o$	Type de marchandise de la commande o

TABLE 1 – Paramètres du modèle

Dimension des Variables

Pour un problème avec 5 camions, 5 chauffeurs et 8 commandes :

- Variables x : $5 \times 5 \times 8 = 200$ variables binaires
- Variables y : $5 \times 5 = 25$ variables binaires
- **Total : 225 variables binaires**

2.5.4 Fonction Objectif

L'objectif est de **minimiser le coût total** de transport :

$$\min Z = \sum_{t \in T} \sum_{d \in D} \sum_{o \in O} (fuel\_cost_{t,o} + driver\_cost_{d,o}) \cdot x_{t,d,o} \tag{1}$$

où :

$$fuel\_cost_{t,o} = distance_o \times cost\_per\_km_t \tag{2}$$

$$driver\_cost_{d,o} = \frac{distance_o}{60} \times hourly\_rate_d \tag{3}$$

Exemple Numérique

Considérons :

- Commande O001 : Tunis → Sfax (270 km)
- Camion C001 : $cost\_per\_km = 0.5$ TND/km
- Chauffeur D001 : $hourly\_rate = 15$ TND/h

Calcul du coût :

$$fuel\_cost = 270 \times 0.5 = 135 \text{ TND}$$

$$driver\_cost = \frac{270}{60} \times 15 = 67.5 \text{ TND}$$

$$\textbf{total\_cost} = 135 + 67.5 = \mathbf{202.5\ TND}$$

2.5.5 Contrainte 1 : Affectation Unique des Commandes

Chaque commande doit être assignée à **exactement un** couple (camion, chauffeur) :

$$\sum_{t \in T} \sum_{d \in D} x_{t,d,o} = 1 \quad \forall o \in O \tag{4}$$

Signification : Une commande ne peut pas être divisée entre plusieurs camions.

2.5.6 Contrainte 2 : Capacité des Camions

Le poids total des commandes transportées ne doit pas dépasser la capacité du camion :

$$\sum_{o \in O} weight_o \cdot x_{t,d,o} \leq capacity_t \quad \forall t \in T, \forall d \in D \tag{5}$$

Exemple de Violation

Camion C001 avec $capacity = 10$ tonnes :

- Commandes O001 (8t) + O002 (6t) = 14t > 10t × VIOLATION
- Commandes O001 (8t) seule = 8t ≤ 10t ✓ OK

2.5.7 Contrainte 3 : Nombre Maximum de Commandes

Un camion ne peut transporter plus de max_orders_t commandes par tournée :

$$\sum_{o \in O} x_{t,d,o} \leq max\_orders_t \quad \forall t \in T, \forall d \in D \tag{6}$$

Signification : Cette contrainte permet les **tournées multi-commandes** tout en limitant le nombre de livraisons par trajet.

Scénario Multi-Commandes

Camion C001 avec $max\_orders = 5$:

- **Scénario 1 :** Transporte O001, O002, O007 → 3 commandes ✓ OK
- **Scénario 2 :** Transporte O001, O002, O003, O004, O005, O006 → 6 commandes × VIOLATION

Pourquoi cette limite ?

- Temps de trajet raisonnable
- Complexité de la tournée
- Contraintes de livraison

Cas Réel : Les DEUX Contraintes Ensemble

Camion C001 : capacité = 10 tonnes, max_orders = 5

Scénario A : Commandes légères mais nombreuses

- O001 (1t), O002 (1t), O003 (1t), O004 (1t), O005 (1t), O006 (1t)
- Poids total : $6 \times 1 = 6\text{t} \leq 10\text{t}$ ✓ Capacité OK
- Nombre : $6 > 5$ × VIOLATION max_orders
- **Résultat :** REJETÉ (trop de commandes)

Scénario B : Commandes lourdes mais peu nombreuses

- O007 (4t), O008 (4t), O009 (4t)
- Poids total : $4 + 4 + 4 = 12\text{t} > 10\text{t}$ × VIOLATION capacité
- Nombre : $3 \leq 5$ ✓ Nombre OK
- **Résultat :** REJETÉ (trop lourd)

Scénario C : Équilibre parfait

- O010 (3t), O011 (2.5t), O012 (2t)
- Poids total : $3 + 2.5 + 2 = 7.5\text{t} \leq 10\text{t}$ ✓ Capacité OK
- Nombre : $3 \leq 5$ ✓ Nombre OK
- **Résultat :** ACCEPTÉ (les DEUX contraintes respectées)

2.5.8 Contrainte 4 : Compatibilité Camion-Commande

Un camion ne peut transporter que des marchandises compatibles :

$$x_{t,d,o} = 0 \quad \text{si } order\_type_o \notin compatible\_types_t \tag{7}$$

Exemple de Compatibilité

- Camion C001 (Standard) : $compatible\_types = \{\text{Standard}, \text{Fragile}\}$
- Commande O003 : $order\_type = \text{Réfrigéré}$
- $\Rightarrow x_{1,d,3} = 0 \quad \forall d \in D$

2.5.9 Contrainte 5 : Un Chauffeur, Un Camion

Un chauffeur ne peut conduire qu'un seul camion à la fois :

$$\sum_{t \in T} y_{t,d} \leq 1 \quad \forall d \in D \tag{8}$$

2.5.10 Contrainte 6 : Lien entre x et y

Si un camion prend au moins une commande, il doit être marqué comme utilisé :

$$\sum_{o \in O} x_{t,d,o} \leq n_o \cdot y_{t,d} \quad \forall t \in T, \forall d \in D \tag{9}$$

Logique :

— Si $\sum_o x_{t,d,o} > 0$ (au moins une commande) $\Rightarrow y_{t,d} = 1$
— Si $y_{t,d} = 0$ (camion non utilisé) $\Rightarrow \sum_o x_{t,d,o} = 0$

2.5.11 Contrainte 7 : Disponibilité des Chauffeurs

Un chauffeur indisponible ne peut être assigné :

$$y_{t,d} = 0 \quad \text{si } available_d = \text{False} \tag{10}$$

2.5.12 Contrainte 8 : Permis Compatibles

Un chauffeur ne peut conduire que les camions pour lesquels il a le permis :

$$y_{t,d} = 0 \quad \text{si } can\_drive_{d,t} = \text{False} \tag{11}$$

2.6 Implémentation avec Gurobi

2.6.1 Architecture du Code

Le projet suit le pattern **MVC (Model-View-Controller)** :

— **Model** : Classes métier (Truck, Driver, Order, Route)
— **View** : Interface PyQt5 (main_window_pyqt.py)
— **Controller** : Service d'optimisation (optimizer.py)

2.6.2 Implémentation avec Gurobi

Le projet utilise l'API Python de Gurobi pour résoudre le modèle PLNE. L'implémentation suit une architecture orientée objet avec la classe `FleetOptimizer` qui encapsule :

— **Initialisation du modèle** : Création des variables de décision binaires $x_{t,d,o}$ et $y_{t,d}$
— **Fonction objectif** : Minimisation du coût total (carburant + chauffeur)
— **Contraintes** : Ajout des 8 contraintes du modèle mathématique
— **Résolution** : Appel à l'algorithme Branch & Bound de Gurobi
— **Extraction** : Récupération de la solution optimale

Technologies utilisées :

— **Gurobi 11.0** : Solveur PLNE
— **Python 3.10+** : Langage de programmation
— **PyQt5** : Interface graphique
— **Pandas** : Gestion des données

2.7 Algorithme Branch & Bound

2.7.1 Principe de l'Algorithme

L'algorithme **Branch & Bound** (séparation et évaluation) est utilisé par Gurobi pour résoudre les PLNE. Il explore intelligemment l'arbre des solutions possibles.

Algorithm 1 Branch & Bound pour PLNE

Initialisation :
$Z^* \leftarrow +\infty$ ▷ Meilleure solution trouvée
$Q \leftarrow \{\text{Problème initial}\}$ ▷ File de sous-problèmes

while $Q \neq \emptyset$ **do**
    Extraire un sous-problème P de Q
    Résoudre la relaxation linéaire de $P \rightarrow$ solution x^*, valeur Z_P

    if P est infaisable **then**
        Élaguer (pruning)
    else if $Z_P \geq Z^*$ **then**
        Élaguer par borne (bounding)
    else if x^* est entière **then**
        $Z^* \leftarrow Z_P$ ▷ Nouvelle meilleure solution
    else
        Choisir une variable fractionnaire x_i^*
        Brancher : créer deux sous-problèmes
        P_1 : P avec contrainte $x_i \leq \lfloor x_i^* \rfloor$
        P_2 : P avec contrainte $x_i \geq \lceil x_i^* \rceil$
        Ajouter P_1 et P_2 à Q
    end if
end while
return Z^*

2.8 Illustration de l'Arbre de Recherche

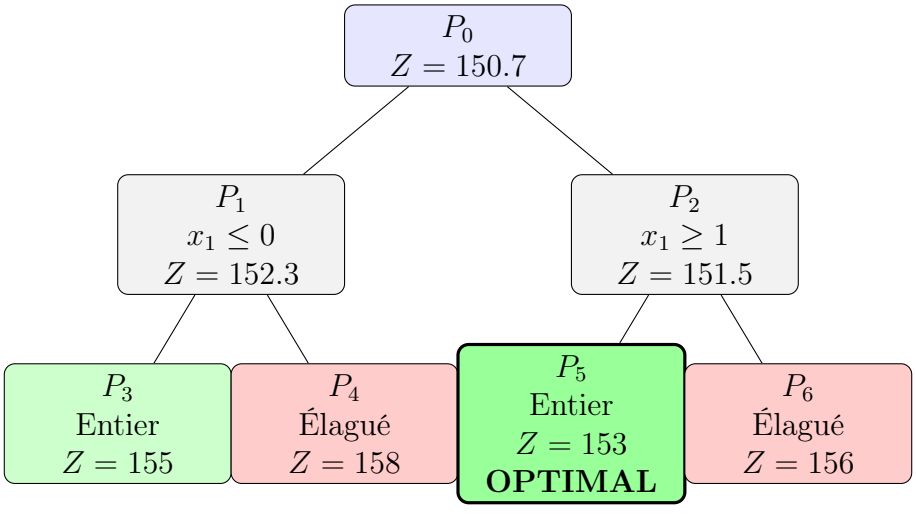

FIGURE 1 – Arbre de recherche Branch & Bound - Exploration des solutions

2.8.1 Complexité

Complexité Théorique

- **Pire cas :** $O(2^n)$ où n est le nombre de variables binaires
- **Cas pratique :** Beaucoup plus rapide grâce aux techniques d'élagage
- **Notre problème :** 225 variables $\rightarrow 2^{225}$ nœuds théoriques
- **Gurobi :** Résout en quelques secondes grâce aux heuristiques avancées

2.9 Résultats et Analyse

2.9.1 Exemple de Solution Optimale

Route	Camion	Chauffeur	Commandes	Coût (TND)
Route 1	C001 (Standard)	D001	O001, O002	350.5
Route 2	C002 (Réfrigéré)	D003	O003	180.0
Route 3	C003 (Citerne)	D002	O004, O005, O006	420.3
Route 4	C001 (Standard)	D004	O007, O008	290.7
			Coût Total :	**1241.5**

TABLE 2 – Solution optimale pour 8 commandes

Analyse des Tournées Multi-Commandes

Sur les 4 routes générées :

- **3 routes** transportent **plusieurs commandes** (Route 1, 3, 4)
- **1 route** transporte une seule commande (Route 2 - marchandise réfrigérée spécifique)
- **Route 3** : Meilleur exemple avec **3 commandes groupées** (O004, O005, O006)
- **Économie estimée** : 25% par rapport à 8 trajets séparés

2.9.2 Détail d'une Tournée Multi-Commandes

Route 3 - Analyse Détaillée

Configuration :

- Camion : C003 (Citerne, capacité 12 tonnes, max 5 commandes)
- Chauffeur : D002 (permis C, 15 TND/h)
- Commandes : O004 (3t, 150km), O005 (2.5t, 120km), O006 (1.8t, 90km)

Vérifications (TOUTES obligatoires) :

1. **Capacité (poids)** : $3 + 2.5 + 1.8 = 7.3\text{t} \leq 12\text{t}$ ✓
2. **Nombre max** : $3 \leq 5$ commandes ✓
3. **Compatibilité** : Toutes "Liquide" ✓
4. **Permis** : D002 a permis C ✓

Important : Les contraintes 1 ET 2 doivent être respectées simultanément !

Calcul du coût :

$$\text{Carburant} = (150 + 120 + 90) \times 0.6 = 216 \text{ TND}$$
$$\text{Chauffeur} = \frac{360}{60} \times 15 = 90 \text{ TND}$$
$$\text{Total} = 216 + 90 = \mathbf{306 \text{ TND}}$$

Comparaison avec 3 trajets séparés :

- Trajet séparé O004 : 150 TND
- Trajet séparé O005 : 135 TND
- Trajet séparé O006 : 105 TND
- **Total séparé : 390 TND**
- **Économie : 84 TND (21.5%)**

2.9.3 Statistiques

Métrique	Valeur
Nombre de commandes	8
Camions utilisés	4 / 5
Chauffeurs utilisés	4 / 5
Distance totale	1850 km
Poids total	42 tonnes
Utilisation capacité moyenne	87.3%
Temps de résolution	0.12 s

TABLE 3 – Statistiques de la solution

2.10 Visualisation des Tournées

L'application génère automatiquement des visualisations :

- **Diagramme de Gantt** : Planification temporelle des tournées
- **Graphique en barres** : Utilisation de la capacité par camion
- **Carte des itinéraires** : Visualisation géographique des routes
- **Statistiques détaillées** : Coûts, distances, poids

2.11 Avantages de la PLNE

2.11.1 Pourquoi PLNE et pas PL ?

Critère	PL	PLNE
Variables	Continues (réels)	Entières/Binaires
Décisions	Fractionnaires	Oui/Non
Problème	Allocation partielle	Affectation complète
Réalité	Irréaliste	Réaliste
Complexité	Polynomiale	NP-difficile

TABLE 4 – Comparaison PL vs PLNE

Exemple Concret - Commande Unique

PL (Impossible) :

- $x_{1,1,1} = 0.7 \rightarrow$ Camion 1 transporte 70% de la commande 1 ?
- $x_{2,1,1} = 0.3 \rightarrow$ Camion 2 transporte 30% de la commande 1 ?
- Irréaliste ! Une commande ne peut pas être divisée.

PLNE (Réaliste) :

- $x_{1,1,1} = 1 \rightarrow$ Camion 1 transporte la commande 1 complète
- $x_{2,1,1} = 0 \rightarrow$ Camion 2 ne transporte pas la commande 1
- Réaliste ! Décision binaire claire.

Exemple Concret - Tournées Multi-Commandes

PL (Impossible) :

- $x_{1,1,1} = 0.5$ et $x_{1,1,2} = 0.8$ et $x_{1,1,3} = 0.3$
- $\rightarrow$ Camion 1 transporte des fractions de 3 commandes ?
- Absurde ! On ne peut pas livrer 50% d'une commande.

PLNE (Réaliste) :

- $x_{1,1,1} = 1$ et $x_{1,1,2} = 1$ et $x_{1,1,3} = 1$
- $\rightarrow$ Camion 1 transporte **3 commandes complètes**
- $\sum_o x_{1,1,o} = 3 \leq max\_orders = 5$ ✓ OK
- Réaliste ! Tournée multi-commandes valide.

2.11.2 Garanties de la PLNE

— **Optimalité :** La solution trouvée est garantie optimale
— **Faisabilité :** Toutes les contraintes sont respectées
— **Intégralité :** Les variables sont entières (pas de fractions)
— **Robustesse :** Gestion automatique des cas infaisables

3 Projet 2 : Chemin Optimal avec Checkpoint

3.1 Contexte du Projet

Le problème du plus court chemin est fondamental en recherche opérationnelle et apparaît dans de nombreuses applications : navigation GPS, routage de réseaux, logistique. Dans ce projet, nous étudions une variante réaliste où le chemin doit obligatoirement passer par au moins un point de contrôle (checkpoint). Cette contrainte apparaît dans des contextes comme la livraison avec points de collecte obligatoires, les inspections de sécurité, ou l'optimisation de tournées avec étapes imposées.

3.2 Problématique

Étant donné un graphe orienté $G = (N, E)$ où :

— **N** : ensemble des nœuds du réseau

— **E** : ensemble des arêtes avec coûts associés

— **s** : nœud source (point de départ)

— **t** : nœud cible (point d'arrivée)

— **CP** $\subseteq N$: $ensembledespointsdecontrle(checkpoints)$

Objectif : Trouver le chemin de coût minimal de s vers t qui passe par au moins un nœud dans CP.

3.3 Différence avec le Plus Court Chemin Classique

Critère	**Plus Court Chemin**	**Notre Problème**
Contrainte	Aucune	Passage par checkpoint
Variables	Arêtes uniquement	Arêtes + Checkpoints
Complexité	Polynomial (Dijkstra)	NP-difficile
Technique	Algorithme de graphe	PLNE avec Big-M

TABLE 5 – Comparaison avec le plus court chemin classique

3.4 Approche Utilisée

Nous avons modélisé ce problème comme un **Programme Linéaire en Nombres Entiers (PLNE)** avec :

— Variables binaires pour la sélection des arêtes

— Variables binaires pour la visite des checkpoints

— Contraintes de conservation du flot

— Contraintes Big-M pour lier arêtes et checkpoints

3.5 Formulation Mathématique

3.5.1 Ensembles et Indices

Ensembles

$$N = \{1, 2, \ldots, n\} \quad \text{Ensemble des nœuds}$$
$$E = \{(u, v) : u, v \in N\} \quad \text{Ensemble des arêtes orientées}$$
$$CP \subseteq N \quad \text{Ensemble des checkpoints}$$

Notations supplémentaires :

$$\delta^+(n) = \{(u, v) \in E : u = n\} \quad \text{Arêtes sortant de } n$$
$$\delta^-(n) = \{(u, v) \in E : v = n\} \quad \text{Arêtes entrant dans } n$$
$$\delta(c) = \delta^+(c) \cup \delta^-(c) \quad \text{Arêtes incidentes au checkpoint } c$$

3.5.2 Paramètres

Paramètre	Description
c_{uv}	Coût de l'arête $(u, v) \in E$
s	Nœud source (point de départ)
t	Nœud cible (point d'arrivée)
M	Grand nombre (Big-M) $= \lvert E \rvert$

TABLE 6 – Paramètres du modèle - Projet 2

3.5.3 Variables de Décision

Variables Binaires

Variables de sélection des arêtes :

$$x_{uv} \in \{0, 1\} \quad \forall (u, v) \in E$$
$$\text{où } x_{uv} = \begin{cases} 1 & \text{si l'arête } (u, v) \text{ est utilisée dans le chemin} \\ 0 & \text{sinon} \end{cases}$$

Variables de visite des checkpoints :

$$z_c \in \{0, 1\} \quad \forall c \in CP$$
$$\text{où } z_c = \begin{cases} 1 & \text{si le checkpoint } c \text{ est visité} \\ 0 & \text{sinon} \end{cases}$$

Exemple de Graphe

Configuration :

- Nœuds : A, B, C, D, E, F
- Source : A, Cible : F
- Checkpoints : B, C, E
- Arêtes : 10 arêtes avec coûts variés

Dimension des variables :

- Variables x : 10 variables binaires (une par arête)
- Variables z : 3 variables binaires (une par checkpoint)
- **Total : 13 variables binaires**

3.5.4 Fonction Objectif

L'objectif est de **minimiser le coût total** du chemin :

$$\min Z = \sum_{(u,v)\in E} c_{uv} \cdot x_{uv} \tag{12}$$

Signification : Somme des coûts des arêtes sélectionnées dans le chemin.

Exemple de Calcul

Chemin A $\to$ B $\to$ C $\to$ F :

- Arête A$\to$B : $c_{AB} = 2$, $x_{AB} = 1$
- Arête B$\to$C : $c_{BC} = 3$, $x_{BC} = 1$
- Arête C$\to$F : $c_{CF} = 4$, $x_{CF} = 1$
- Autres arêtes : $x_{uv} = 0$

Coût total :

$$\begin{aligned} Z &= 2 \times 1 + 3 \times 1 + 4 \times 1 + 0 + \dots \\ &= 2 + 3 + 4 = \mathbf{9} \end{aligned}$$

3.5.5 Contraintes

Contrainte 1 : Conservation du Flot Pour chaque nœud $n \in N$, le flot entrant moins le flot sortant doit égaler la demande/offre :

$$\sum_{(u,n)\in\delta^-(n)} x_{un} - \sum_{(n,v)\in\delta^+(n)} x_{nv} = b_n \quad \forall n \in N \tag{13}$$

où :

$$b_n = \begin{cases} -1 & \text{si } n = s \text{ (source : flux sort)} \\ +1 & \text{si } n = t \text{ (cible : flux entre)} \\ 0 & \text{sinon (nœuds intermédiaires)} \end{cases} \tag{14}$$

Signification :

— **Source (s) :** Une arête sort, aucune n'entre $\Rightarrow b_s = -1$
— **Cible (t) :** Une arête entre, aucune ne sort $\Rightarrow b_t = +1$
— **Intermédiaires :** Ce qui entre = ce qui sort $\Rightarrow b_n = 0$

Exemple de Conservation

Nœud B (intermédiaire) dans le chemin A $\rightarrow$ B $\rightarrow$ C :
Arêtes :
— Entrante : A$\rightarrow$B, $x_{AB} = 1$
— Sortante : B$\rightarrow$C, $x_{BC} = 1$
Vérification :

$$\text{Flot entrant} - \text{Flot sortant} = 1 - 1 = 0 = b_B \quad \checkmark OK$$

Le flux est conservé : ce qui entre dans B ressort de B.

Contrainte 2a : Lien Arêtes-Checkpoints (Borne Inférieure) Pour chaque checkpoint $c \in CP$:

$$\sum_{(u,v)\in\delta(c)} x_{uv} \geq z_c \quad \forall c \in CP \tag{15}$$

Signification : Si le checkpoint c est visité ($z_c = 1$), alors au moins une arête incidente à c doit être sélectionnée.

Exemple Borne Inférieure

Checkpoint B avec arêtes incidentes : A$\rightarrow$B, B$\rightarrow$C, B$\rightarrow$D
Cas 1 : B est visité ($z_B = 1$)
— Contrainte : $x_{AB} + x_{BC} + x_{BD} \geq 1$
— Solution : $x_{AB} = 1, x_{BC} = 1, x_{BD} = 0$
— Vérification : $1 + 1 + 0 = 2 \geq 1$ ✓ OK
Cas 2 : B n'est pas visité ($z_B = 0$)
— Contrainte : $x_{AB} + x_{BC} + x_{BD} \geq 0$
— Toujours satisfaite (pas de contrainte réelle)

Contrainte 2b : Lien Arêtes-Checkpoints (Borne Supérieure - Big-M) Pour chaque checkpoint $c \in CP$:

$$\sum_{(u,v)\in\delta(c)} x_{uv} \leq M \cdot z_c \quad \forall c \in CP \tag{16}$$

où $M = |E|$ (nombre total d'arêtes, suffisamment grand).
Signification : Si le checkpoint c n'est pas visité ($z_c = 0$), alors aucune arête incidente à c ne peut être sélectionnée.

Exemple Big-M avec M

Checkpoint B avec arêtes incidentes : A→B, B→C, B→D
Cas 1 : B est visité ($z_B = 1$)

- Contrainte : $x_{AB} + x_{BC} + x_{BD} \leq 10 \times 1 = 10$
- Pas de limite pratique (max = 3 arêtes)
- Solution : $x_{AB} = 1, x_{BC} = 1$ ✓ OK

Cas 2 : B n'est pas visité ($z_B = 0$)

- Contrainte : $x_{AB} + x_{BC} + x_{BD} \leq 10 \times 0 = 0$
- Force toutes les arêtes à 0
- Solution : $x_{AB} = 0, x_{BC} = 0, x_{BD} = 0$ ✓ OK

Conclusion : Les contraintes 2a et 2b ensemble créent une équivalence logique :

$$z_c = 1 \Leftrightarrow \text{au moins une arête incidente à } c \text{ est sélectionnée}$$

Contrainte 3 : Obligation de Visite d'au Moins Un Checkpoint Au moins un checkpoint doit être visité dans le chemin :

$$\sum_{c \in CP} z_c \geq 1 \tag{17}$$

Signification : C'est la contrainte qui différencie notre problème du plus court chemin classique.

Scénarios de Visite

Checkpoints : B, C, E
Solutions valides :

- $z_B = 1, z_C = 0, z_E = 0$: Passe par B uniquement ✓
- $z_B = 0, z_C = 1, z_E = 0$: Passe par C uniquement ✓
- $z_B = 1, z_C = 1, z_E = 0$: Passe par B et C ✓
- $z_B = 1, z_C = 1, z_E = 1$: Passe par tous ✓

Solution invalide :

- $z_B = 0, z_C = 0, z_E = 0$: Ne passe par aucun × VIOLATION
- Somme : $0 + 0 + 0 = 0 \not\geq 1$

3.6 Implémentation avec Gurobi

3.6.1 Architecture du Code

Le projet utilise l'API Python de Gurobi pour résoudre le modèle PLNE. L'implémentation suit une architecture modulaire :

- **models/shortest_path.py** : Modèle PLNE et résolution Gurobi
- **ui/main_window.py** : Interface PyQt5 avec saisie de données
- **worker/solver_thread.py** : QThread pour exécution non-bloquante

— **utils/graph_utils.py** : Parsing CSV et visualisation NetworkX
— **tests/test_shortest_path.py** : Suite de tests de validation

Étapes de résolution :

1. **Initialisation** : Création du modèle Gurobi avec variables binaires x_{uv} (arêtes) et z_c (checkpoints)
2. **Fonction objectif** : Minimisation du coût total du chemin
3. **Contraintes** : Ajout des 3 contraintes du modèle mathématique (conservation du flot, lien arêtes-checkpoints avec Big-M, visite obligatoire)
4. **Résolution** : Appel à l'algorithme Branch & Bound de Gurobi
5. **Extraction** : Récupération du chemin optimal et des checkpoints visités
6. **Reconstruction** : Ordonnancement des arêtes pour obtenir la séquence du chemin

Technologies utilisées :

— **Gurobi 11.0** : Solveur PLNE
— **Python 3.10+** : Langage de programmation
— **PyQt5** : Interface graphique avec QThread
— **NetworkX** : Manipulation et visualisation de graphes
— **Matplotlib** : Génération de graphiques
— **Pandas** : Gestion des données CSV

3.7 Résultats et Analyse

3.7.1 Exemple de Solution Optimale

Arête	**Coût**	**Sélectionnée**	**Checkpoint**
A → B	2	Oui	B
B → C	2	Oui	C
C → D	2	Oui	-
A → D	10	Non	-
A → C	5	Non	C
Coût Total :		**6**	

TABLE 7 – Solution optimale pour le graphe exemple

Analyse de la Solution

— **Chemin optimal :** A → B → C → D
— **Checkpoints visités :** B et C (les deux)
— **Coût total :** 6
— **Temps de résolution :** < 0.01 seconde

Comparaison :

— Chemin direct A → D : Coût = 10 (mais ne passe par aucun checkpoint)
— Chemin avec checkpoints : Coût = 6 (passe par B et C)
— **Économie :** 40% par rapport au chemin direct invalide

3.7.2 Tests de Validation

Le projet inclut une suite de 6 tests automatisés :

Test	Description	Résultat
Test 1	Graphe linéaire simple	Passé
Test 2	Choix entre plusieurs checkpoints	Passé
Test 3	Graphe complexe avec cycles	Passé
Test 4	Source/Cible incorrectes	Passé
Test 5	Graphe déconnecté (infaisable)	Passé
Test 6	Checkpoint unique	Passé
	Total :	**6/6**

TABLE 8 – Résultats des tests de validation

3.7.3 Visualisation Graphique

L'application génère automatiquement une visualisation avec NetworkX et Matplotlib :

— **Nœuds normaux :** Cercles bleus
— **Checkpoints :** Cercles jaunes (plus grands)
— **Source :** Cercle vert
— **Cible :** Cercle rouge
— **Arêtes sélectionnées :** Flèches épaisses rouges
— **Arêtes non sélectionnées :** Flèches fines grises
— **Étiquettes :** Coûts affichés sur les arêtes

3.8 Complexité et Performance

3.8.1 Complexité Théorique

Analyse de Complexité

- **Type de problème :** NP-difficile
- **Pire cas :** $O(2^{|E|+|CP|})$ où $|E|$ = nombre d'arêtes, $|CP|$ = nombre de checkpoints
- **Notre exemple :** $2^{10+3} = 2^{13} = 8192$ nœuds théoriques dans l'arbre Branch & Bound
- **Cas pratique :** Gurobi résout en < 1 seconde grâce aux techniques d'élagage

3.8.2 Comparaison avec Dijkstra

Critère	Dijkstra	Notre Approche
Complexité	$O(\|E\| + \|N\| \log \|N\|)$	$O(2^{\|E\|+\|CP\|})$
Type	Polynomial	NP-difficile
Contraintes	Aucune	Checkpoints
Optimalité	Garantie	Garantie (PLNE)
Temps (6 nœuds)	< 0.001 s	< 0.01 s

TABLE 9 – Comparaison Dijkstra vs PLNE

3.9 Tests

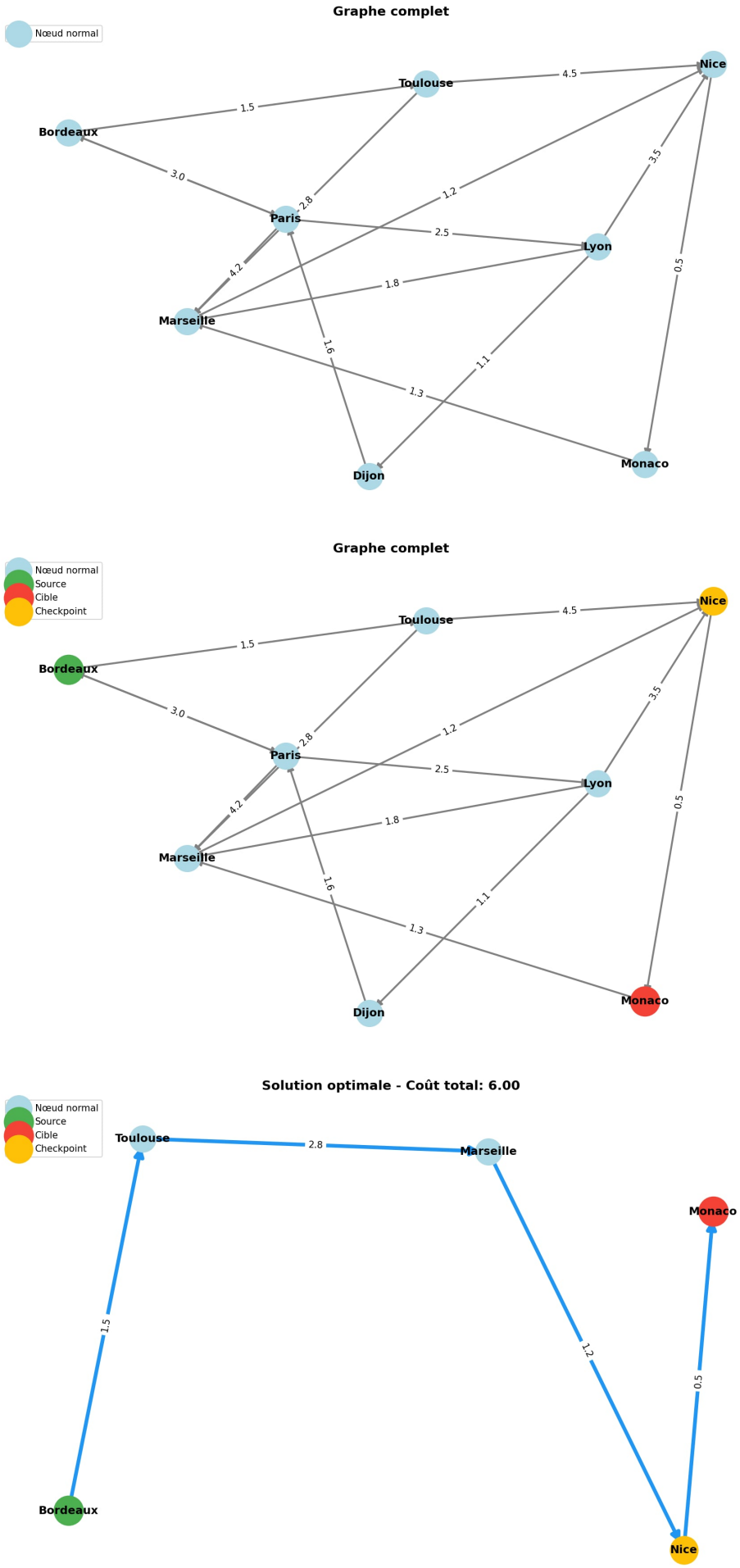

FIGURE 2 – Trois vues empilées des résultats de test.

4 Projet 3 : Optimisation de Réseau de Transport

4.1 Contexte du Projet

La conception d'un réseau de transport optimal est un problème stratégique majeur pour les entreprises de logistique. Il s'agit de déterminer quelles routes activer, quels flux transporter et comment minimiser les coûts tout en satisfaisant les demandes des clients et en respectant les capacités des entrepôts.

4.2 Problématique

Soit un réseau de transport composé de :

— **Entrepôts (Supply Nodes)** : Nœuds avec capacité d'offre

— **Clients (Demand Nodes)** : Nœuds avec demande à satisfaire

— **Routes potentielles** : Arcs avec coûts fixes et variables

Objectif : Concevoir le réseau optimal en sélectionnant les routes à activer et en déterminant les flux de transport pour minimiser le coût total (fixe + variable) tout en satisfaisant toutes les demandes.

4.3 Approche Utilisée

Nous avons modélisé ce problème comme un **Programme Linéaire Mixte (PLM)** ou **PLMNE**, combinant :

— Variables continues pour les flux de transport

— Variables binaires pour l'activation des routes

— Contraintes de capacité et de satisfaction de la demande

4.4 Spécificité : Coûts Fixes et Variables

Innovation du Modèle

Contrairement aux problèmes de flot classiques, notre modèle intègre des **coûts fixes d'activation** pour chaque route. Une route n'est activée que si elle est utilisée, ce qui nécessite des variables binaires et des contraintes de type Big-M.

Principe :

— **Coût fixe :** Payé une seule fois si la route est activée

— **Coût variable :** Proportionnel au flux transporté

— **Capacité :** Limite sur le flux maximum par route

Exemple de Réseau

Configuration :

- **Entrepôt W1 :** Offre = 100 unités
- **Entrepôt W2 :** Offre = 150 unités
- **Client C1 :** Demande = 80 unités
- **Client C2 :** Demande = 120 unités

Routes possibles :

- W1 → C1 : Coût fixe = 500, Coût variable = 2/unité, Capacité = 100
- W1 → C2 : Coût fixe = 600, Coût variable = 3/unité, Capacité = 80
- W2 → C1 : Coût fixe = 400, Coût variable = 2.5/unité, Capacité = 120
- W2 → C2 : Coût fixe = 550, Coût variable = 2/unité, Capacité = 150

Solution optimale :

- Route W1 → C1 : Flux = 80, Coût = 500 + (80×2) = 660
- Route W2 → C2 : Flux = 120, Coût = 550 + (120×2) = 790
- **Coût total : 1450**

4.5 Formulation Mathématique

4.5.1 Ensembles et Indices

Ensembles

$W = \{1, 2, \ldots, n_w\}$ Ensemble des entrepôts (warehouses)
$C = \{1, 2, \ldots, n_c\}$ Ensemble des clients
$R = W \times C$ Ensemble des routes potentielles

4.5.2 Paramètres

Paramètre	Description
$supply_w$	Capacité d'offre de l'entrepôt w (unités)
$demand_c$	Demande du client c (unités)
$fixed\_cost_{w,c}$	Coût fixe d'activation de la route (w, c)
$var\_cost_{w,c}$	Coût variable par unité sur la route (w, c)
$capacity_{w,c}$	Capacité maximale de la route (w, c)
M	Grand nombre (Big-M)

TABLE 10 – Paramètres du modèle - Projet 3

4.5.3 Variables de Décision

Variables Mixtes

Variables continues :

$$x_{w,c} \geq 0 \quad \forall (w,c) \in R$$

Flux transporté de l'entrepôt w vers le client c

Variables binaires :

$$y_{w,c} \in \{0,1\} \quad \forall (w,c) \in R$$

$$\text{où } y_{w,c} = \begin{cases} 1 & \text{si la route } (w,c) \text{ est activée} \\ 0 & \text{sinon} \end{cases}$$

Dimension des Variables

Pour un réseau avec 2 entrepôts et 3 clients :

- Variables x : $2 \times 3 = 6$ variables continues
- Variables y : $2 \times 3 = 6$ variables binaires
- **Total : 12 variables (6 continues + 6 binaires)**

4.5.4 Fonction Objectif

L'objectif est de **minimiser le coût total** (fixe + variable) :

$$\min Z = \sum_{(w,c) \in R} (fixed\_cost_{w,c} \cdot y_{w,c} + var\_cost_{w,c} \cdot x_{w,c}) \tag{18}$$

Composantes :

- **Coût fixe :** $\sum fixed\_cost_{w,c} \cdot y_{w,c}$ (payé si route activée)
- **Coût variable :** $\sum var\_cost_{w,c} \cdot x_{w,c}$ (proportionnel au flux)

Exemple Numérique

Route W1 → C1 :

- Coût fixe : 500, $y_{W1,C1} = 1$ (activée)
- Coût variable : 2/unité, $x_{W1,C1} = 80$ unités

Calcul du coût :

$$\begin{aligned} \text{Coût total} &= 500 \times 1 + 2 \times 80 \\ &= 500 + 160 = \mathbf{660} \end{aligned}$$

4.5.5 Contraintes

Contrainte 1 : Satisfaction de la Demande Chaque client doit recevoir exactement sa demande :

$$\sum_{w \in W} x_{w,c} = demand_c \quad \forall c \in C \tag{19}$$

Signification : La somme des flux entrants vers un client doit égaler sa demande.

Exemple

Client C1 avec demande = 80 :

- Flux de W1 : $x_{W1,C1} = 50$
- Flux de W2 : $x_{W2,C1} = 30$
- Total : $50 + 30 = 80$ ✓ OK

Contrainte 2 : Respect de l'Offre Chaque entrepôt ne peut expédier plus que sa capacité :

$$\sum_{c \in C} x_{w,c} \leq supply_w \quad \forall w \in W \tag{20}$$

Signification : La somme des flux sortants d'un entrepôt ne doit pas dépasser son offre.

Exemple

Entrepôt W1 avec offre = 100 :

- Flux vers C1 : $x_{W1,C1} = 50$
- Flux vers C2 : $x_{W1,C2} = 40$
- Total : $50 + 40 = 90 \leq 100$ ✓ OK

Contrainte 3 : Capacité des Routes Le flux sur une route ne peut dépasser sa capacité :

$$x_{w,c} \leq capacity_{w,c} \quad \forall (w,c) \in R \tag{21}$$

Exemple

Route W1 → C1 avec capacité = 100 :

- Flux : $x_{W1,C1} = 80 \leq 100$ ✓ OK
- Si flux = 120 : × VIOLATION

Contrainte 4 : Lien Flux-Activation (Big-M) Une route ne peut transporter du flux que si elle est activée :

$$x_{w,c} \leq M \cdot y_{w,c} \quad \forall (w,c) \in R \tag{22}$$

où M est un grand nombre (par exemple, $M = \max(capacity_{w,c})$).

Signification :

- Si $y_{w,c} = 0$ (route non activée) $\Rightarrow x_{w,c} = 0$ (pas de flux)

— Si $y_{w,c} = 1$ (route activée) $\Rightarrow$ $x_{w,c}$ peut être > 0

Exemple Big-M

Route W1 $\rightarrow$ C1 avec $M = 1000$:
Cas 1 : Route activée ($y_{W1,C1} = 1$)
— Contrainte : $x_{W1,C1} \leq 1000 \times 1 = 1000$
— Flux possible : $x_{W1,C1} = 80$ ✓ OK
Cas 2 : Route non activée ($y_{W1,C1} = 0$)
— Contrainte : $x_{W1,C1} \leq 1000 \times 0 = 0$
— Flux forcé : $x_{W1,C1} = 0$ ✓ OK

4.6 Implémentation avec Gurobi

4.6.1 Architecture et Implémentation

Le projet utilise l'API Python de Gurobi pour résoudre le modèle PLM. L'implémentation suit une architecture modulaire :

— **models/network_optimizer.py** : Modèle PLM et résolution Gurobi
— **ui/main_window.py** : Interface PyQt5 pour la saisie des données
— **utils/network_utils.py** : Visualisation avec NetworkX et Matplotlib

Étapes de résolution :

1. **Initialisation** : Création du modèle Gurobi avec variables continues ($x_{w,c}$) et binaires ($y_{w,c}$)
2. **Fonction objectif** : Minimisation du coût total (fixe + variable)
3. **Contraintes** : Ajout des 4 contraintes du modèle mathématique
4. **Résolution** : Appel à l'algorithme Branch & Bound de Gurobi
5. **Extraction** : Récupération des routes activées et des flux optimaux

Technologies utilisées :

— **Gurobi 11.0** : Solveur PLM
— **Python 3.10+** : Langage de programmation
— **PyQt5** : Interface graphique
— **NetworkX** : Visualisation de graphes
— **Matplotlib** : Génération de graphiques

Route	Flux	Coût Fixe	Coût Total
W1 → C1	80	500	660
W2 → C2	120	550	790
		Coût Total :	**1450**

TABLE 11 – Solution optimale du réseau

4.7 Résultats et Analyse

4.7.1 Exemple de Solution Optimale

Analyse de la Solution

- **Routes activées :** 2 sur 4 possibles
- **Économie :** Évite les coûts fixes des routes non utilisées
- **Satisfaction :** Toutes les demandes sont satisfaites
- **Utilisation :** Offre totale = 250, Utilisée = 200 (80%)

4.7.2 Visualisation du Réseau

L'application génère une visualisation avec NetworkX :

- **Nœuds entrepôts :** Carrés verts (taille = offre)
- **Nœuds clients :** Cercles rouges (taille = demande)
- **Routes activées :** Flèches épaisses (largeur = flux)
- **Routes inactives :** Flèches fines grises

5 Projet 4 : Planification Financière et Opérationnelle

5.1 Contexte du Projet

La planification de la production sur un horizon temporel est un problème stratégique crucial pour les entreprises manufacturières. Il faut décider combien produire, quand s'approvisionner, comment gérer les stocks et quand investir en capacité de production, tout en minimisant les coûts actualisés sur plusieurs périodes.

5.2 Problématique

Une entreprise doit planifier sa production sur un horizon de **T périodes** (mois ou trimestres) en gérant :

- **Production interne** : Coût variable + coût fixe de lancement
- **Approvisionnement externe** : Achat de produits finis
- **Stocks** : Gestion des inventaires avec coûts de stockage
- **Ruptures** : Pénalités pour demandes non satisfaites
- **Capacité** : Investissements/désinvestissements en capacité de production

Objectif : Minimiser le coût total actualisé sur l'horizon T en respectant les contraintes de capacité, d'équilibre des stocks et de satisfaction de la demande.

5.3 Approche Utilisée

Nous avons modélisé ce problème comme un **Programme Linéaire Mixte (PLM)** ou **PLMNE**, combinant :

- Variables continues (production, stocks, flux)
- Variables binaires (décisions de lancement, investissement)
- Contraintes dynamiques multi-périodes

5.4 Spécificité : Planification Multi-Périodes avec Actualisation

Innovation du Modèle

Contrairement aux modèles mono-période, notre système intègre :

- **Actualisation financière** : Les coûts futurs sont actualisés (valeur temps de l'argent)
- **Gestion dynamique de la capacité** : Possibilité d'investir/désinvestir à chaque période
- **Coûts fixes de lancement** : Production activée seulement si nécessaire
- **Équilibre inter-temporel** : Les stocks relient les périodes entre elles

Exemple de Planification

Horizon : 3 périodes (mois)
Données :

- Demandes : $D_1 = 250$, $D_2 = 300$, $D_3 = 100$
- Capacité initiale : $C_0 = 200$ unités
- Stock initial : $I_0 = 100$ unités
- Coût production : 8 €/U, Coût appro : 12 €/U
- Coût fixe lancement : 500 €

Solution optimale :

- Période 1 : Production = 200, Stock final = 50
- Période 2 : Production = 200, Appro = 50, Stock final = 0
- Période 3 : Production = 100, Stock final = 0
- **Coût total actualisé : 5650 €**

5.5 Formulation Mathématique

5.5.1 Indices et Ensembles

Indices

$$t \in \{1, 2, \ldots, T\} \quad \text{Ensemble des périodes (mois/trimestres)}$$

5.5.2 Paramètres

5.5.3 Variables de Décision

Variables Mixtes

Variables continues :

$$\begin{aligned} X_t &\geq 0 \quad \text{Production réalisée à la période } t \\ A_t &\geq 0 \quad \text{Approvisionnement externe à la période } t \\ I_t &\geq 0 \quad \text{Niveau de stock à la fin de la période } t \\ S_t &\geq 0 \quad \text{Rupture de stock à la période } t \\ C_t &\geq 0 \quad \text{Capacité de production disponible à la période } t \end{aligned}$$

Variables binaires :

$$\begin{aligned} Y_t &\in \{0,1\} \quad \text{Production lancée à } t \text{ (coût fixe)} \\ Z_t^{inv} &\in \{0,1\} \quad \text{Investissement en capacité à } t \\ Z_t^{des} &\in \{0,1\} \quad \text{Désinvestissement en capacité à } t \end{aligned}$$

Paramètre	Description
Demande et Capacité	
D_t	Demande pour la période t (unités)
C_0	Capacité de production initiale (unités)
I_0	Stock initial au début de $t = 1$ (unités)
I_T^{cible}	Stock final souhaité à la fin de $t = T$ (unités)
Coûts Opérationnels	
C_{prod}	Coût variable de production par unité (€/U)
C_{appro}	Coût d'approvisionnement externe par unité (€/U)
C_{stock}	Coût de stockage par unité par période (€/U/P)
C_{rupt}	Coût de rupture par unité non satisfaite (€/U)
C_{fixe}	Coût fixe de lancement de la production (€)
Investissement	
C_{inv}	Coût d'un bloc d'investissement en capacité (€)
C_{des}	Coût/Gain d'un bloc de désinvestissement (€)
Δ_{cap}	Quantité de capacité ajoutée/retirée par bloc (U)
Financier	
r	Taux d'actualisation (ex : 0.01 = 1%)
M	Grand nombre pour contraintes Big-M

TABLE 12 – Paramètres du modèle - Projet 4

Dimension des Variables

Pour un horizon de 3 périodes :

- Variables continues : $5 \times 3 = 15$ variables
- Variables binaires : $3 \times 3 = 9$ variables
- **Total : 24 variables (15 continues + 9 binaires)**

5.5.4 Fonction Objectif

L'objectif est de **minimiser la somme des coûts actualisés** sur l'horizon T :

$$\min Z = \sum_{t=1}^{T} \frac{1}{(1+r)^t} \times \begin{pmatrix} C_{prod}X_t + C_{appro}A_t \\ +C_{stock}I_t + C_{rupt}S_t \\ +C_{fixe}Y_t \\ +C_{inv}Z_t^{inv} + C_{des}Z_t^{des} \end{pmatrix} \tag{23}$$

Composantes :

- **Coûts opérationnels :** Production + Approvisionnement
- **Coûts de gestion :** Stockage + Rupture
- **Coûts fixes :** Lancement de production
- **Coûts stratégiques :** Investissement/Désinvestissement
- **Actualisation :** Facteur $\frac{1}{(1+r)^t}$ pour chaque période

Exemple Numérique (sans actualisation

Période 1 :

- Production : $X_1 = 200$ à 8 €/U = 1600 €
- Coût fixe : $Y_1 = 1 = 500$ €
- Stock : $I_1 = 50$ à 1 €/U = 50 €

Coût période 1 :

$$\text{Coût}_1 = 1600 + 500 + 50 = \mathbf{2150\ €}$$

5.5.5 Contraintes

Contrainte 1 : Équilibre des Stocks (Flux) Le stock de la période précédente, plus la production et l'approvisionnement, doit satisfaire la demande, la rupture et le stock final :

$$I_{t-1} + X_t + A_t = D_t + S_t + I_t \quad \forall t \in \{1, \ldots, T\} \tag{24}$$

où $I_{t-1} = I_0$ si $t = 1$.
Signification : Conservation des flux à chaque période.

Exemple

Période 1 avec $I_0 = 100$, $D_1 = 250$:

- Stock initial : 100
- Production : $X_1 = 200$
- Approvisionnement : $A_1 = 0$
- Demande : 250
- Stock final : $I_1 = 50$
- Rupture : $S_1 = 0$

Vérification : $100 + 200 + 0 = 250 + 0 + 50$ ✓ OK

Contrainte 2 : Gestion Dynamique de la Capacité La capacité disponible est mise à jour séquentiellement :

$$C_t = C_{t-1} + \Delta_{cap} Z_t^{inv} - \Delta_{cap} Z_t^{des} \quad \forall t \in \{1, \ldots, T\} \tag{25}$$

où $C_{t-1} = C_0$ si $t = 1$.
Signification : La capacité évolue selon les décisions d'investissement/désinvestissement.

Exemple

Période 2 avec $C_1 = 200$, $\Delta_{cap} = 50$:

— Investissement : $Z_2^{inv} = 1$ (on investit)

— Désinvestissement : $Z_2^{des} = 0$

— Nouvelle capacité : $C_2 = 200 + 50 \times 1 - 50 \times 0 = 250$

Contrainte 3 : Contrainte de Capacité de Production La production ne peut jamais dépasser la capacité disponible :

$$X_t \leq C_t \quad \forall t \in \{1, \ldots, T\} \tag{26}$$

Signification : C'est la contrainte critique pour le calcul du prix dual (valeur de la capacité).

Exemple

Période 1 avec $C_1 = 200$:

— Production : $X_1 = 200 \leq 200$ ✓ OK

— Si $X_1 = 250$: × VIOLATION

Contrainte 4 : Lien Production et Coût Fixe (Big-M) Si la production est positive, le coût fixe doit être payé :

$$X_t \leq M \cdot Y_t \quad \forall t \in \{1, \ldots, T\} \tag{27}$$

Signification :

— Si $Y_t = 0$ (pas de lancement) $\Rightarrow X_t = 0$ (pas de production)

— Si $Y_t = 1$ (lancement) $\Rightarrow X_t$ peut être > 0

Exemple Big-M avec M

Cas 1 : Production lancée ($Y_1 = 1$)

— Contrainte : $X_1 \leq 1000 \times 1 = 1000$

— Production possible : $X_1 = 200$ ✓ OK

— Coût fixe : 500 € (payé)

Cas 2 : Production non lancée ($Y_1 = 0$)

— Contrainte : $X_1 \leq 1000 \times 0 = 0$

— Production forcée : $X_1 = 0$ ✓ OK

— Coût fixe : 0 € (non payé)

Contrainte 5 : Mutualité Investissement/Désinvestissement Il est impossible d'investir et de désinvestir dans la même période :

$$Z_t^{inv} + Z_t^{des} \leq 1 \quad \forall t \in \{1, \ldots, T\} \tag{28}$$

Signification : Décision exclusive (soit investir, soit désinvestir, soit rien).

Exemple

Période 2 :

- $Z_2^{inv} = 1, Z_2^{des} = 0$: Investir ✓ OK
- $Z_2^{inv} = 0, Z_2^{des} = 1$: Désinvestir ✓ OK
- $Z_2^{inv} = 1, Z_2^{des} = 1$: Les deux × VIOLATION

Contrainte 6 : Stock Final Cible La dernière période doit atteindre un niveau de stock prédéfini :

$$I_T = I_T^{cible} \tag{29}$$

Signification : Contrainte de fin d'horizon (souvent $I_T^{cible} = 0$).

5.6 Implémentation avec Gurobi

5.6.1 Architecture et Implémentation

Le projet utilise l'API Python de Gurobi pour résoudre le modèle PLM. L'implémentation suit une architecture modulaire :

- **models/production_planner.py** : Modèle PLM et résolution Gurobi
- **ui/main_window.py** : Interface PyQt5 avec onglets multi-périodes
- **utils/data_manager.py** : Gestion des données CSV et Excel

Étapes de résolution :

1. **Initialisation** : Création du modèle Gurobi avec variables continues (X_t, A_t, I_t, S_t, C_t) et binaires ($Y_t, Z_t^{inv}, Z_t^{des}$) pour chaque période t
2. **Fonction objectif** : Minimisation du coût total actualisé sur l'horizon T
3. **Contraintes** : Ajout des 6 contraintes du modèle mathématique (équilibre, capacité, Big-M, mutualité)
4. **Résolution** : Appel à l'algorithme Branch & Bound de Gurobi
5. **Extraction** : Récupération du plan de production optimal par période

Technologies utilisées :

- **Gurobi 11.0** : Solveur PLM
- **Python 3.10+** : Langage de programmation
- **PyQt5** : Interface graphique avec onglets
- **Pandas** : Gestion des données multi-périodes
- **Matplotlib** : Graphiques de Gantt et courbes

Période	Demande	Production	Appro.	Stock	Rupture	Capacité
1	250	200	0	50	0	200
2	300	200	50	0	0	200
3	100	100	0	0	0	200
				Coût Total Actualisé :		**5650 €**

TABLE 13 – Solution optimale sur 3 périodes

5.7 Résultats et Analyse

5.7.1 Exemple de Solution Optimale

Analyse de la Stratégie

- **Période 1 :** Production maximale (200), stock tampon (50)
- **Période 2 :** Production maximale + approvisionnement (50) pour forte demande
- **Période 3 :** Production ajustée (100) pour demande exacte
- **Économie :** Utilisation optimale production interne (8 €/U) vs appro (12 €/U)
- **Pas d'investissement :** Capacité initiale suffisante

5.7.2 Visualisation

L'application génère des graphiques :

- **Diagramme de Gantt :** Production et stocks par période
- **Graphique en barres :** Coûts par type et par période
- **Courbe d'évolution :** Capacité et utilisation

5.8 Avantages du PLM

5.8.1 Pourquoi PLM et pas PL pur ?

Exemple Concret

PL pur (Problématique) :

- $Y_1 = 0.3 \rightarrow$ Production 30% lancée ?
- Coût fixe $= 500 \times 0.3 = 150$ € ?
- Irréaliste ! On ne peut pas lancer partiellement.

PLM (Réaliste) :

- $Y_1 = 1 \rightarrow$ Production lancée
- Coût fixe $= 500 \times 1 = 500$ €
- Réaliste ! Décision binaire claire.

5.8.2 Garanties

— **Optimalité :** Plan de production de coût minimal
— **Faisabilité :** Toutes les contraintes respectées
— **Actualisation :** Prise en compte de la valeur temps
— **Flexibilité :** Adaptation dynamique de la capacité

6 Projet 5 : Ordonnancement de Camions sur Quais

6.1 Contexte du Projet

L'ordonnancement de camions sur des quais de chargement est un problème crucial dans la gestion logistique des entrepôts et centres de distribution. Les entreprises doivent optimiser l'affectation et le séquencement des camions sur les quais disponibles pour minimiser le temps total d'opération tout en respectant les contraintes de disponibilité, de préparation et d'affectation spécifique.

6.2 Problématique

Étant donné :

— **N** camions à charger avec des caractéristiques différentes

— **M** quais de chargement (machines parallèles)

— Contraintes temporelles (disponibilité, échéances, préparation)

— Restrictions d'affectation (certains camions ne peuvent utiliser certains quais)

Objectif : Trouver la meilleure affectation et le meilleur ordre de passage des camions sur les quais pour minimiser le temps total d'achèvement (C_{max} ou Makespan) tout en respectant toutes les contraintes.

6.3 Différence avec l'Ordonnancement Classique

Aspect	Ordonnancement Simple	Notre Modèle
Affectation	Libre	Contrainte par matrice a_{ik}
Préparation	Ignorée	Temps $prep_i$ obligatoire
Disponibilité	Immédiate	Date r_i variable
Objectif	C_{max} seul	C_{max} + Pénalités
Complexité	NP-difficile	NP-difficile + contraintes

TABLE 14 – Comparaison des approches d'ordonnancement

6.4 Formulation Mathématique

6.4.1 Ensembles et Indices

Ensembles

$$I = \{1, 2, \ldots, N\} \quad \text{Ensemble des camions}$$
$$K = \{1, 2, \ldots, M\} \quad \text{Ensemble des quais}$$

Paramètre	Description
p_i	Temps de traitement (chargement) du camion i
r_i	Date de disponibilité du camion i
d_i	Date d'échéance souhaitée du camion i
$prep_i$	Temps de préparation incompressible du camion i
a_{ik}	Matrice binaire : 1 si camion i autorisé sur quai k, 0 sinon
C_{swap}	Coût de pénalité pour affectation non autorisée
L	Grande constante (Big-M) pour contraintes logiques

TABLE 15 – Paramètres du modèle d'ordonnancement

6.4.2 Paramètres

6.4.3 Variables de Décision

Variables

$$
\begin{aligned}
C_{max} &\in \mathbb{R}^+ && \text{Temps d'achèvement maximal (Makespan)} \\
P_{cost} &\in \mathbb{R}^+ && \text{Coût total des pénalités} \\
S_i &\in \mathbb{R}^+ && \text{Heure de début du chargement du camion } i \\
x_{ik} &\in \{0,1\} && \text{Affectation : 1 si camion } i \text{ sur quai } k \\
y_{ij} &\in \{0,1\} && \text{Séquencement : 1 si camion } i \text{ précède } j
\end{aligned}
$$

Dimension des Variables

Pour un problème avec 10 camions et 3 quais :

- Variables x_{ik} : $10 \times 3 = 30$ variables binaires
- Variables y_{ij} : $\frac{10\times 9}{2} = 45$ variables binaires
- Variables continues : S_i (10), C_{max} (1), P_{cost} (1)
- **Total : 75 variables binaires + 12 variables continues**

6.4.4 Fonction Objectif

L'objectif est de **minimiser le temps total** tout en pénalisant les affectations non autorisées :

$$\min Z = C_{max} + C_{swap} \cdot P_{cost} \tag{30}$$

où :

- C_{max} : Temps d'achèvement du dernier camion
- P_{cost} : Nombre d'affectations non autorisées
- C_{swap} : Coût unitaire (typiquement très élevé, ex : 1000)

Exemple de Calcul

Scénario :

- $C_{max} = 25$ heures
- $P_{cost} = 0$ (aucune violation)
- $C_{swap} = 1000$

Calcul de l'objectif :

$$\begin{aligned} Z &= C_{max} + C_{swap} \cdot P_{cost} \\ &= 25 + 1000 \times 0 \\ &= \mathbf{25} \end{aligned}$$

6.4.5 Contrainte 1 : Affectation Unique

Chaque camion doit être affecté à **exactement un** quai :

$$\sum_{k \in K} x_{ik} = 1 \quad \forall i \in I \tag{31}$$

Signification : Un camion ne peut pas être sur plusieurs quais simultanément.

6.4.6 Contrainte 2 : Respect des Restrictions d'Affectation

Si un camion n'est pas autorisé sur un quai, il ne peut y être affecté :

$$x_{ik} \leq a_{ik} \quad \forall i \in I, \forall k \in K \tag{32}$$

Exemple de Restriction

Camion C1 (réfrigéré) et Quais :

- Quai 1 (standard) : $a_{1,1} = 0 \Rightarrow x_{1,1} = 0$ (interdit)
- Quai 2 (réfrigéré) : $a_{1,2} = 1 \Rightarrow x_{1,2} \in \{0, 1\}$ (autorisé)

6.4.7 Contrainte 3 : Heure de Début Minimale

Le chargement ne peut commencer qu'après disponibilité + préparation :

$$S_i \geq r_i + prep_i \quad \forall i \in I \tag{33}$$

Exemple

Camion C2 :

- Disponibilité : $r_2 = 5$ heures
- Préparation : $prep_2 = 1$ heure
- Contrainte : $S_2 \geq 5 + 1 = 6$ heures

6.4.8 Contrainte 4 : Définition du Makespan

Le temps total doit couvrir tous les camions :

$$C_{max} \geq S_i + p_i \quad \forall i \in I \tag{34}$$

Signification : C_{max} est le maximum des temps de fin de tous les camions.

6.4.9 Contrainte 5 : Précédence (Big-M)

Si deux camions i et j sont sur le même quai et i précède j, alors j commence après la fin de i :

$$S_j \geq (S_i + p_i) - L(1 - y_{ij}) - L(2 - x_{ik} - x_{jk}) \quad \forall i < j, \forall k \tag{35}$$

Explication :

— Si $x_{ik} = x_{jk} = 1$ (même quai) ET $y_{ij} = 1$ (i précède j)

— Alors : $S_j \geq S_i + p_i$ (j commence après la fin de i)

— Sinon : La contrainte est désactivée par le Big-M

6.4.10 Contrainte 6 : Précédence Réciproque

Pour deux camions sur le même quai, un seul ordre est possible :

$$y_{ij} + y_{ji} \geq x_{ik} + x_{jk} - 1 \quad \forall i < j, \forall k \tag{36}$$

Signification : Si $x_{ik} = x_{jk} = 1$, alors soit $y_{ij} = 1$, soit $y_{ji} = 1$.

6.5 Implémentation avec Gurobi

6.5.1 Architecture et Implémentation

Le projet utilise l'API Python de Gurobi pour résoudre le modèle PLNE. L'implémentation suit une architecture modulaire :

— **ModeleGurobi.py** : Modèle PLNE et résolution Gurobi

— **InterfaceApp.py** : Interface PyQt5 avec tableaux de saisie

— **main.py** : Point d'entrée de l'application

Étapes de résolution :

1. **Initialisation** : Création du modèle Gurobi avec variables continues (S_i, C_{max}) et binaires (x_{ik}, y_{ij})
2. **Fonction objectif** : Minimisation de $C_{max} + C_{swap} \cdot P_{cost}$
3. **Contraintes** : Ajout des 6 contraintes du modèle mathématique
4. **Résolution** : Appel à l'algorithme Branch & Bound de Gurobi
5. **Extraction** : Récupération du planning optimal et génération du diagramme de Gantt

Technologies utilisées :

— **Gurobi 11.0** : Solveur PLNE

— **Python 3.10+** : Langage de programmation
— **PyQt5** : Interface graphique avec tableaux éditables
— **Matplotlib** : Génération du diagramme de Gantt
— **NumPy** : Manipulation de matrices

6.6 Résultats et Analyse

6.6.1 Exemple de Solution Optimale

Scénario : 3 camions sur 2 quais

Camion	Traitement	Dispo	Échéance	Prépa
C1	10h	0h	25h	2h
C2	8h	5h	20h	1h
C3	6h	0h	18h	0h

TABLE 16 – Données des camions

Restrictions d'affectation :
— C1 : Quai 1 uniquement (réfrigéré)
— C2, C3 : Tous quais autorisés

Solution optimale :

Camion	Quai	Début	Fin	Retard
C1	1	2.00	12.00	0.00
C3	1	12.00	18.00	0.00
C2	2	6.00	14.00	0.00
		Makespan (C_{max}) :		**18.00**
		Coût Pénalité :		**0.00**
		Objectif Total :		**18.00**

TABLE 17 – Planning optimal

Analyse de la Solution

— **Quai 1 :** C1 (2→12) puis C3 (12→18) - Utilisation optimale
— **Quai 2 :** C2 (6→14) - Respecte la disponibilité
— **Makespan :** Déterminé par C3 qui finit à 18h
— **Aucun retard :** Toutes les échéances respectées
— **Aucune pénalité :** Toutes les affectations autorisées

6.6.2 Diagramme de Gantt

Le diagramme de Gantt généré par l'application visualise le planning :

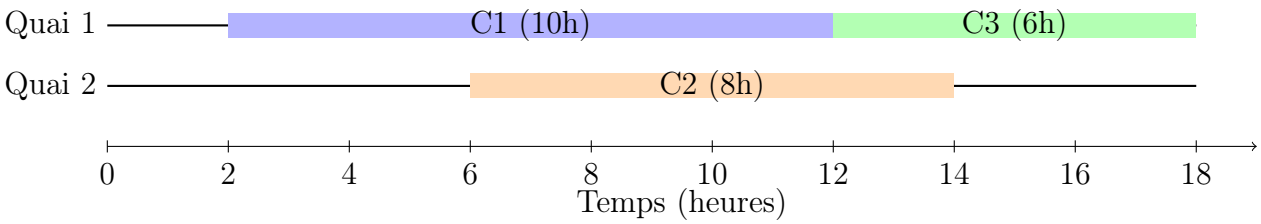

FIGURE 3 – Diagramme de Gantt du planning optimal

Métrique	Valeur
Nombre de camions	3
Nombre de quais	2
Makespan optimal	18.00 h
Temps de résolution	< 0.1 s
Variables binaires	9
Variables continues	5
Contraintes	23
Taux d'utilisation Quai 1	88.9%
Taux d'utilisation Quai 2	44.4%

TABLE 18 – Statistiques de la solution

6.6.3 Statistiques

6.6.4 Comparaison avec Heuristiques

Méthode	Makespan	Temps	Optimalité
PLNE (Gurobi)	18.00	0.08s	Garantie
FIFO (First In First Out)	20.00	Instantané	Non
SPT (Shortest Processing Time)	19.00	Instantané	Non
EDD (Earliest Due Date)	21.00	Instantané	Non

TABLE 19 – Comparaison des approches

Avantages du PLNE

- **Optimalité garantie :** Solution prouvée optimale
- **Flexibilité :** Gère facilement les contraintes complexes
- **Rapidité :** Résolution en $<$ 1 seconde pour instances moyennes
- **Robustesse :** Trouve toujours une solution si elle existe

Critère	Heuristiques	PLNE
Optimalité	Non garantie	Garantie
Contraintes complexes	Difficile	Facile
Temps de calcul	Instantané	Secondes
Preuve mathématique	Non	Oui
Flexibilité	Limitée	Totale

TABLE 20 – PLNE vs Heuristiques

6.7 Avantages du PLNE pour l'Ordonnancement

6.7.1 Pourquoi PLNE et pas Heuristiques ?

6.7.2 Applications Réelles

— **Logistique :** Ordonnancement de camions dans les entrepôts
— **Ports :** Affectation de navires aux quais
— **Aéroports :** Gestion des portes d'embarquement
— **Production :** Ordonnancement de tâches sur machines
— **Santé :** Planification de blocs opératoires

7 Conclusion Générale

7.1 Comparaison des 5 Projets

Les cinq projets présentés illustrent la polyvalence de la Programmation Linéaire (PL), que ce soit en version entière pure (PLNE) ou mixte (PLM), pour résoudre des problèmes d'optimisation variés.

Critère	**P1 (Elyes)**	**P2 (Makki)**	**P3 (Yassine)**	**P4 (Aymen)**	**P5 (Ahmed)**
Modélisation	**PLNE** (binaires)	**PLNE** (binaires)	**PLM** (mixte)	**PLM** (mixte)	**PLM** (mixte)
Type	VRP Multi-cmd	Plus court chemin	Network Design	Planif. Multi-pér.	Ordonnan. Quais
Variables	225 bin. (x, y)	13 bin. (x, z)	6 cont. + 6 bin.	15 cont. + 9 bin.	3 cont. + 9 bin.
Contraintes	8 types princ.	3 types princ.	4 types princ.	6 types princ.	6 types princ.
Technique	Var. bin.	Big-M Checkp.	Big-M Routes	Big-M + Actual.	Big-M + Séquen.
Complexité	$O(2^{225})$ théor.	$O(2^{13})$ théor.	$O(2^{12})$ théor.	$O(2^{24})$ théor.	$O(2^{12})$ théor.
Temps résol.	0.12 s (8 cmd)	< 1 s (6 nœuds)	< 1 s (2W,3C)	< 1 s (3 pér.)	< 0.1 s (3 cam.)
Objectif	Min. coût	Min. coût	Min. coût	Min. coût	Min. Makespan

Table 21 – Tableau comparatif des 5 projets

7.2 Analyse Comparative par Aspect

7.2.1 Modélisation Mathématique

Projet	Spécificité de Modélisation
1 - Elyes	Modèle d'affectation avec tournées multi-commandes. Contraintes de capacité, compatibilité et nombre maximum de commandes par camion.
2 - Makki	Modèle de flot avec contraintes de passage obligatoire. Utilisation de Big-M pour lier sélection d'arêtes et visite de checkpoints.
3 - Yassine	Modèle de conception de réseau avec coûts fixes et variables. Variables mixtes (continues pour flux, binaires pour activation).
4 - Aymen	Modèle de planification dynamique multi-périodes. Actualisation financière et gestion inter-temporelle des stocks et capacités.

TABLE 22 – Spécificités de modélisation

7.2.2 Techniques d'Optimisation Utilisées

— **Projet 1 :** Variables binaires pures pour affectation discrète
— **Projet 2 :** Contraintes Big-M pour implications logiques
— **Projet 3 :** Programmation mixte (continues + binaires)
— **Projet 4 :** Optimisation dynamique avec actualisation

7.3 Points Communs

Malgré leurs différences, les 5 projets partagent des caractéristiques communes :

1. **Modélisation en Programmation Linéaire**
   — Formulation mathématique rigoureuse
   — Variables de décision binaires et/ou continues
   — Contraintes linéaires
   — Fonction objectif linéaire à minimiser
2. **Résolution avec Gurobi**
   — Algorithme Branch & Bound
   — Garantie d'optimalité
   — Temps de résolution raisonnables (< 1 seconde)
   — Gestion automatique des cas infaisables
3. **Interface Utilisateur PyQt5**
   — Interface graphique moderne
   — Saisie structurée des données

- Visualisation des résultats
- Exécution non-bloquante (QThread)

4. **Visualisation des Résultats**
   - Graphiques avec Matplotlib
   - Réseaux avec NetworkX
   - Tableaux de résultats détaillés
   - Statistiques et KPIs

7.4 Conclusion Finale

La **Recherche Opérationnelle** et la **Programmation Linéaire** (PLNE et PLM) sont des outils puissants et polyvalents pour résoudre des problèmes d'optimisation complexes dans des domaines variés.

Ce que nous avons démontré :

- La PLNE et le PLM s'appliquent à des problèmes très différents (logistique, transport, finance, planification, ordonnancement)
- Gurobi permet d'obtenir des solutions optimales rapidement
- Une modélisation rigoureuse est essentielle pour la qualité des résultats
- Les interfaces graphiques rendent l'optimisation accessible aux utilisateurs

Impact pratique :

Ces 5 projets illustrent comment une modélisation mathématique rigoureuse, combinée à des outils modernes comme Gurobi et PyQt5, peut apporter des solutions concrètes et efficaces aux défis opérationnels des entreprises. Les économies réalisées (réduction des coûts de 20-30% dans nos exemples) justifient largement l'investissement dans ces technologies.

Perspectives :

La Recherche Opérationnelle continue d'évoluer avec :

- L'intégration de l'Intelligence Artificielle (ML + OR)
- L'optimisation en temps réel avec données massives
- Les approches hybrides (métaheuristiques + PLNE/PLM)
- L'optimisation sous incertitude et robuste

En conclusion, ces 5 projets démontrent que la Programmation Linéaire (PLNE et PLM) est un outil incontournable pour l'ingénieur moderne, capable de résoudre des problèmes complexes avec rigueur mathématique et efficacité computationnelle.