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

1. **Projet 1 - Gestion de Flotte (Elyes)** : VRP avec tournées multi-commandes, 225 variables binaires
2. **Projet 2 - Chemin Optimal avec Checkpoint (Makki)** : Plus court chemin avec contraintes de passage
3. **Projet 3 - Réseau de Transport (Yassine)** : Network Design Problem
4. **Projet 4 - Planification Financière (Aymen)** : Planification multi-périodes
5. **Projet 5 - Ordonnancement de Camions (Ahmed)** : Ordonnancement sur machines parallèles

1.3 Objectifs Communs

Modélisation PLNE rigoureuse, résolution optimale avec Gurobi, interface PyQt5, visualisation des résultats.

1.4 Interface Commune de Lancement

Les 5 projets sont accessibles via une **interface de lancement unifiée** développée en PyQt5. Cette interface permet de sélectionner et lancer n'importe quel projet de manière intuitive.

1.4.1 Launcher Principal

Caractéristiques du Launcher :

— **Design moderne** : Interface PyQt5 avec cartes pour chaque projet
— **Navigation intuitive** : Sélection visuelle avec descriptions des fonctionnalités
— **Lancement indépendant** : Chaque projet s'exécute dans son propre processus
— **Architecture modulaire** : 5 projets autonomes avec leurs propres interfaces
— **Responsive** : Adaptation automatique à la taille de l'écran

Projets disponibles :

1. **Projet 1 - Elyes** : Gestion de Flotte Avancée (VRP)
2. **Projet 2 - Makki** : Plus Court Chemin avec Checkpoint
3. **Projet 3 - Yassine** : Réseau de Transport Optimal
4. **Projet 4 - Aymen** : Planification Financière Multi-périodes
5. **Projet 5 - Ahmed** : Ordonnancement de Camions sur Quais

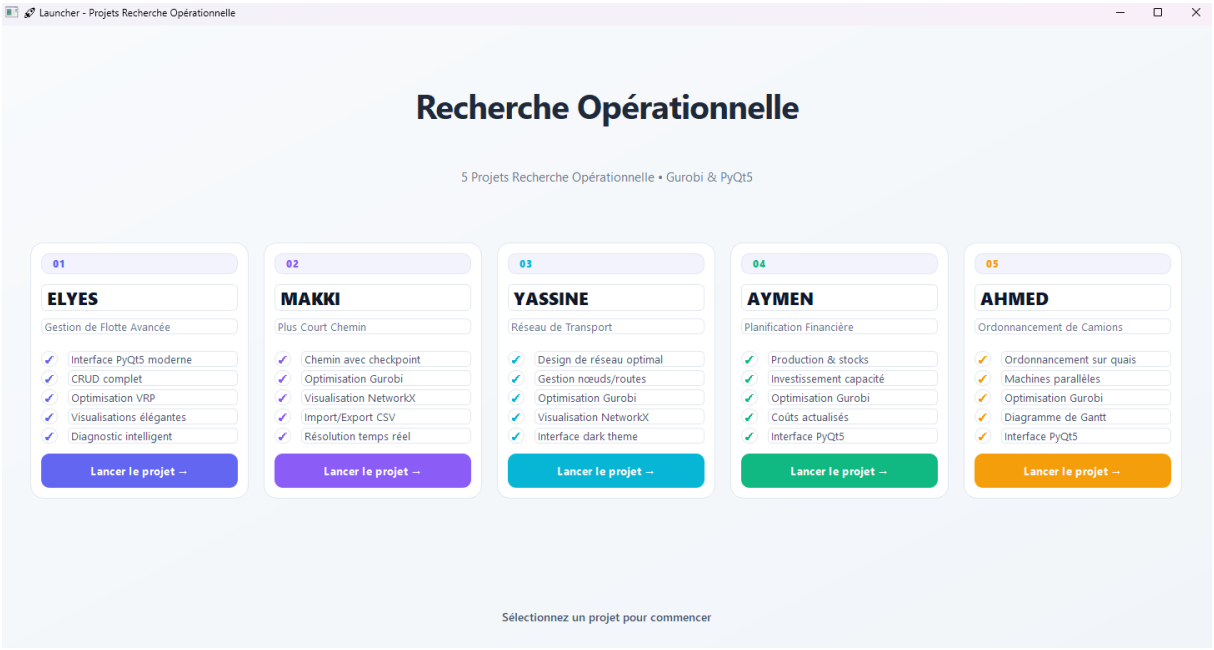

FIGURE 1 – Interface de lancement commune - Sélection des 5 projets de Recherche Opérationnelle

2 Projet 1 : Gestion de Flotte

2.1 Problématique

Une entreprise de transport dispose de $\mathbf{n_t}$ camions, $\mathbf{n_d}$ chauffeurs et doit traiter $\mathbf{n_o}$ commandes. **Objectif :** Affecter les commandes aux couples (camion, chauffeur) en minimisant le coût total.

2.2 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 (jusqu'à max_orders commandes), ce qui optimise l'utilisation des ressources et réduit les coûts.

2.3 Formulation Mathématique

2.3.1 Ensembles et Indices

Ensembles

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

2.3.2 Paramètres

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 maximum 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)
$can\_drive_{d,t}$	Le chauffeur d peut conduire le 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

2.3.3 Variables de Décision

Variables Binaires

Variable principale d'affectation :

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

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

Variable auxiliaire d'activation :

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

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

Justification : Pourquoi x ET y ?

Pourquoi deux variables au lieu d'une seule ?
Rôle de $x_{t,d,o}$:

- Affectation des **commandes** (niveau opérationnel)
- Utilisée dans la fonction objectif et les contraintes C1-C4

Rôle de $y_{t,d}$:

- Activation des **couples (camion, chauffeur)** (niveau stratégique)
- Utilisée dans les contraintes C5-C8 (disponibilité, permis)
- Évite la redondance : $y_{t,d} = 0$ bloque automatiquement toutes les commandes pour ce couple

Avantage : Sans y, il faudrait créer des contraintes redondantes pour chaque combinaison (t, d, o) au lieu d'une seule contrainte par couple (t, d).

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**

2.3.4 Fonction Objectif

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

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

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

2.3.5 Contraintes

C1 - Affectation unique des commandes :
Chaque commande doit être assignée à **exactement un** couple (camion, chauffeur) :

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

C2 - Capacité des camions :
Le poids total des commandes ne doit pas dépasser la capacité :

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

C3 - Nombre maximum de commandes :
Un camion ne peut pas transporter plus de max_orders_t commandes par tournée :

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

C4 - Compatibilité camion-commande :

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

C5 - Un chauffeur, un camion :

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

C6 - Lien entre x et y :
Si un camion transporte au moins une commande, il doit être marqué comme utilisé :

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

C7 - Disponibilité des chauffeurs :

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

C8 - Permis compatibles :

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

2.4 Implémentation avec Gurobi

2.4.1 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)

2.4.2 Technologies Utilisées

— **Gurobi 10.0+** : Solveur PLNE
— **Python 3.10+** : Langage de programmation
— **PyQt5** : Interface graphique
— **Pandas** : Gestion des données

2.5 Algorithme Branch & Bound

2.5.1 Principe de l'Algorithme

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

1. Résolution de la relaxation linéaire du problème
2. Si la solution est entière : mise à jour de l'optimum
3. Si la solution est fractionnaire : séparation en deux sous-problèmes
4. Élagage des branches non prometteuses (*pruning* et *bounding*)

2.5.2 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

2.6 Visualisation des Résultats

L'application génère automatiquement des visualisations :

— **Cartes de statistiques** : 6 cartes visuelles avec les métriques clés (coût, distance, camions utilisés, utilisation de la capacité, commandes)
— **Cartes des tournées** : Visualisation détaillée de chaque tournée avec les itinéraires et les statistiques
— **Graphique en barres** : Comparaison des distances par tournée avec Matplotlib
— **Détails textuels** : Résultats complets au format texte

2.7 Captures d'Écran de l'Interface

2.7.1 Interface Principale

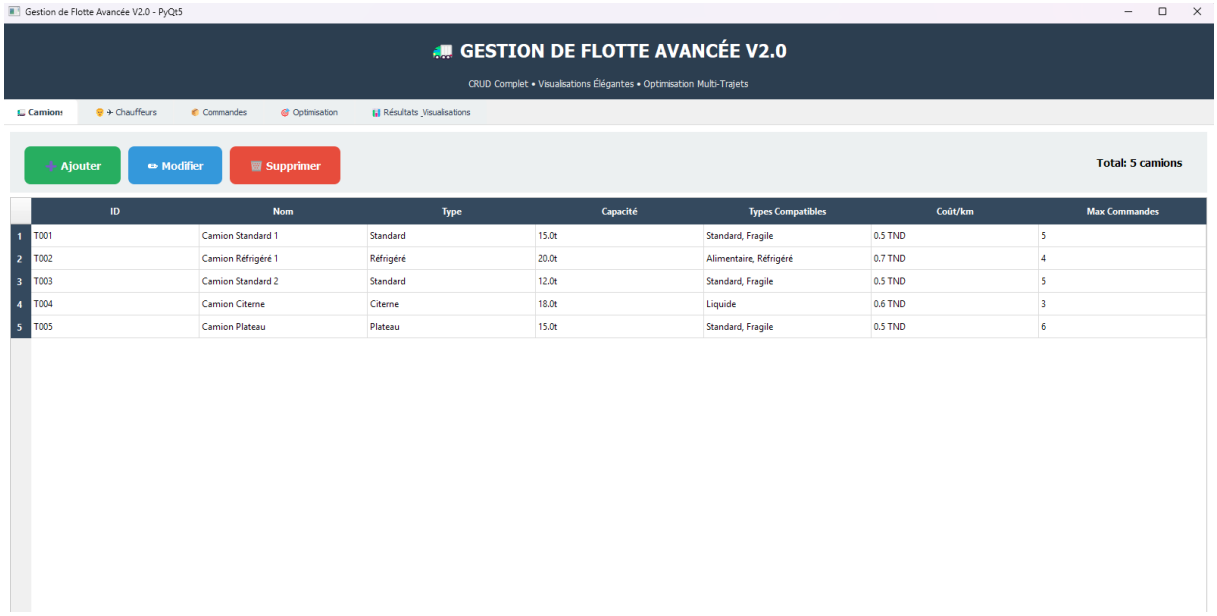

FIGURE 2 – Interface principale de l'application avec gestion des camions, chauffeurs et commandes

2.7.2 Résultats d'Optimisation

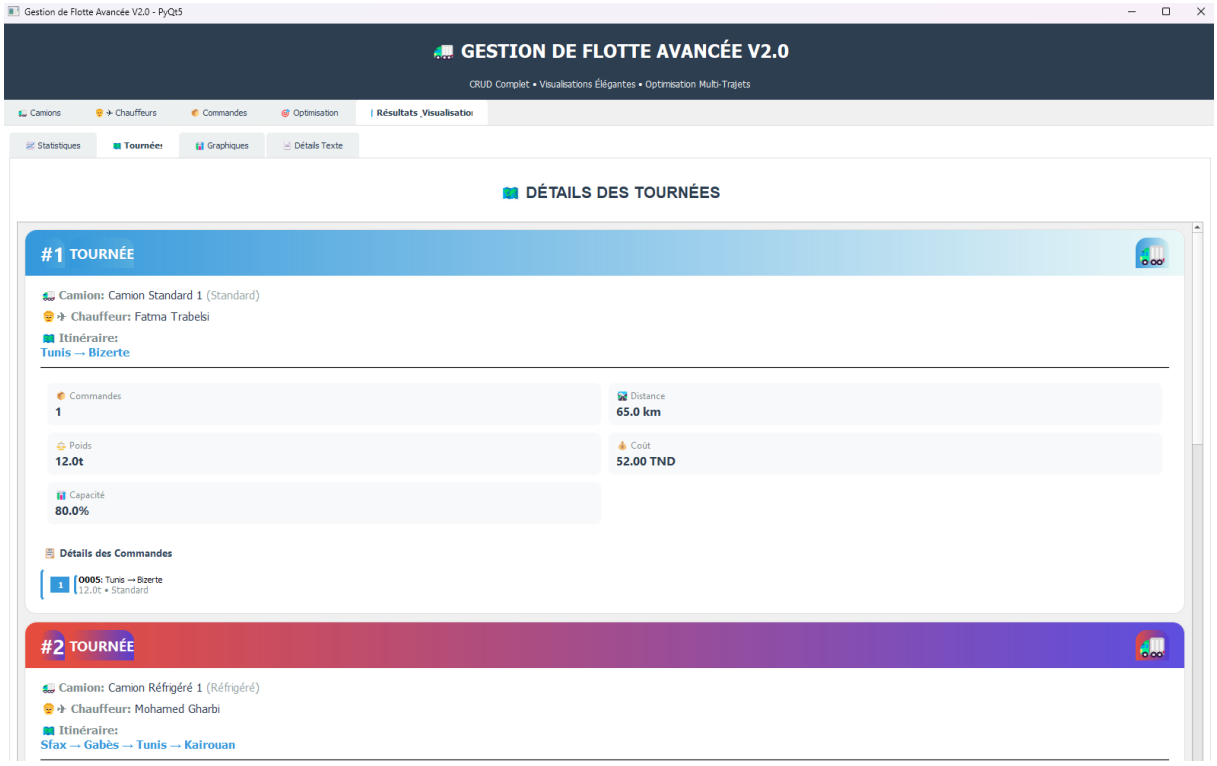

FIGURE 3 – Affichage des résultats d'optimisation avec les tournées assignées

2.7.3 Visualisations

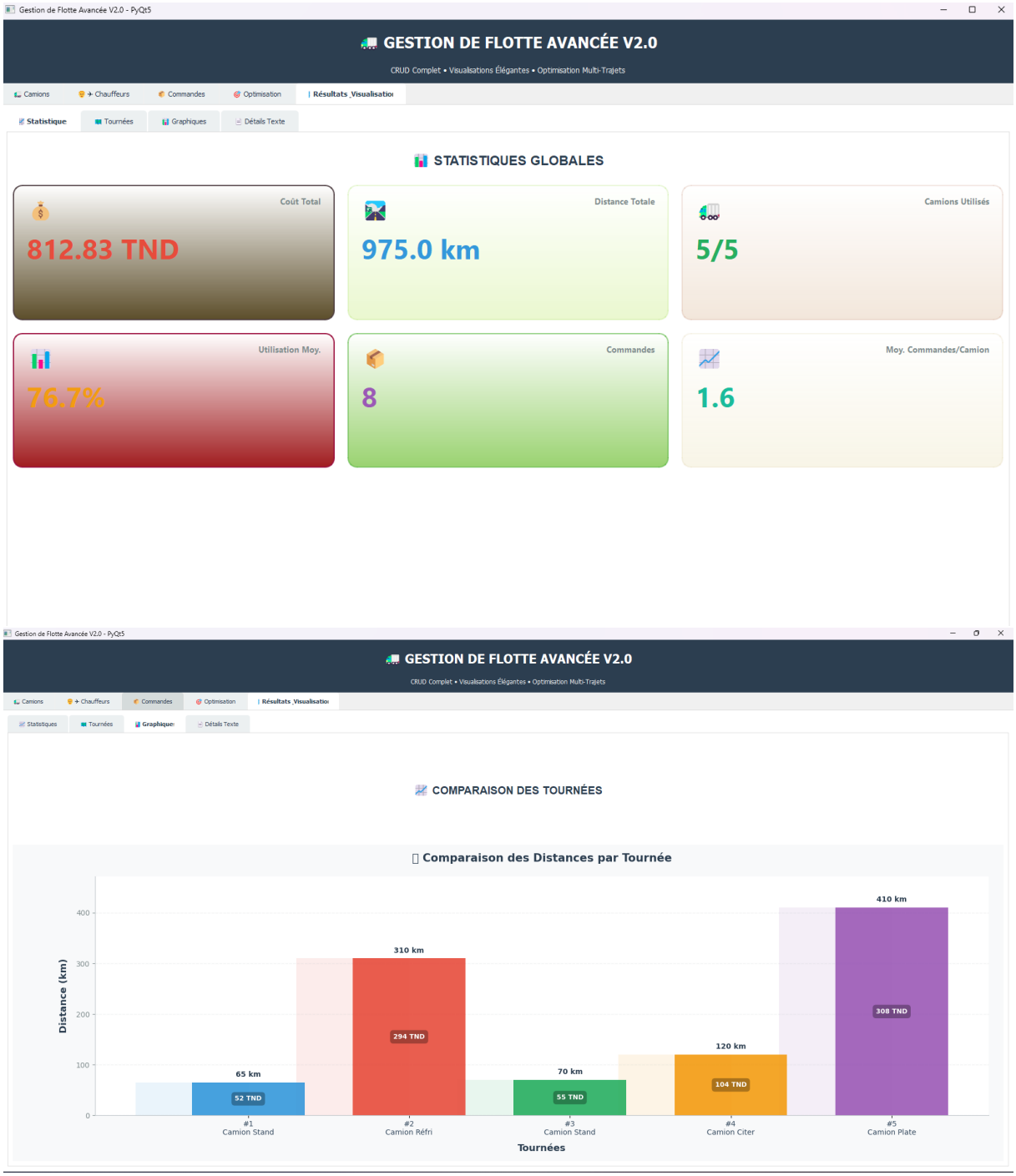

FIGURE 4 – Cartes de statistiques (haut) et graphiques comparatifs (bas)

2.8 Avantages de la PLNE

2.8.1 Pourquoi PLNE et pas PL ?

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

TABLE 2 – Comparaison PL vs PLNE

2.8.2 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. 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 : ensemble des points de contrôle (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

3.4 Approche Utilisée

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

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 3 – Comparaison avec le plus court chemin classique

— 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 4 – 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 :

$$z_c \in \{0,1\} \quad \forall c \in CP$$

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

3.5.4 Fonction Objectif

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

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

3.5.5 Contraintes

C1 - 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 :

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

C2a - 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.

C2b - 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).

Signification : Si le checkpoint c n'est pas visité ($z_c = 0$), alors aucune arête incidente à c ne peut être sélectionnée.

Équivalence Logique

Les contraintes C2a et C2b ensemble créent une équivalence :

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

C3 - 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.

3.6 Implémentation avec Gurobi

3.6.1 Architecture du Code

Le projet utilise 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

3.6.2 Étapes de Résolution

1. **Initialisation** : Création du modèle avec variables binaires x_{uv} et z_c
2. **Fonction objectif** : Minimisation du coût total du chemin
3. **Contraintes** : Ajout des 3 contraintes (conservation du flot, lien arêtes-checkpoints, 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

3.6.3 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 Visualisation et Complexité

3.7.1 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

3.7.2 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
— **Cas pratique :** Gurobi résout efficacement grâce aux techniques d'élagage

3.7.3 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)

TABLE 5 – Comparaison Dijkstra vs PLNE

3.8 Captures d'Écran de l'Interface

3.8.1 Interface de Saisie

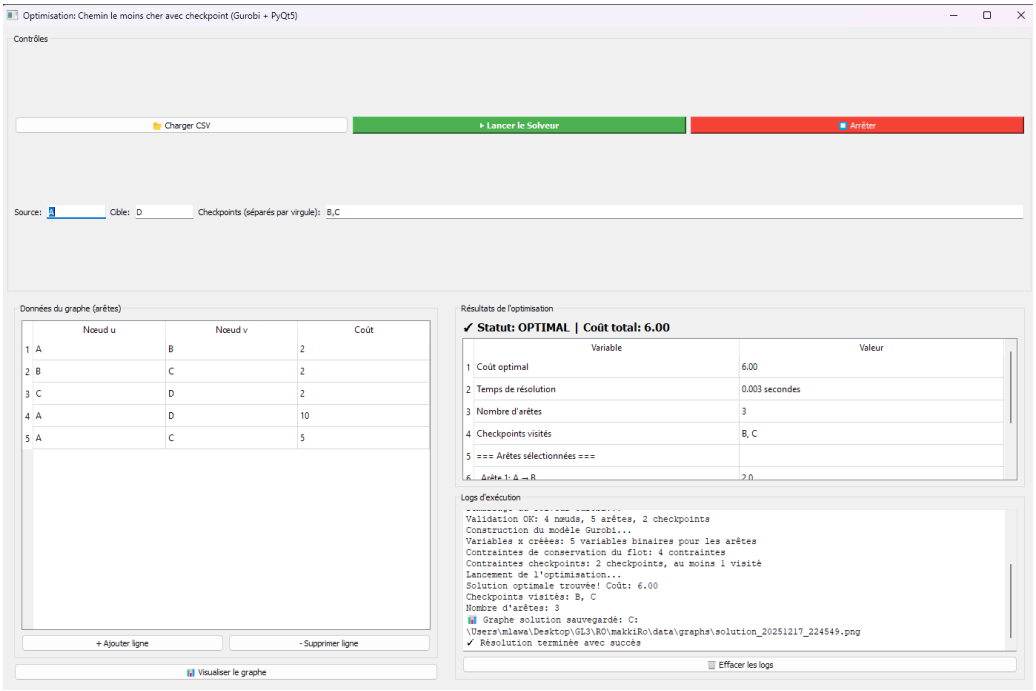

FIGURE 5 – Interface de saisie du graphe avec nœuds, arêtes et checkpoints

3.8.2 Visualisation du Chemin Optimal

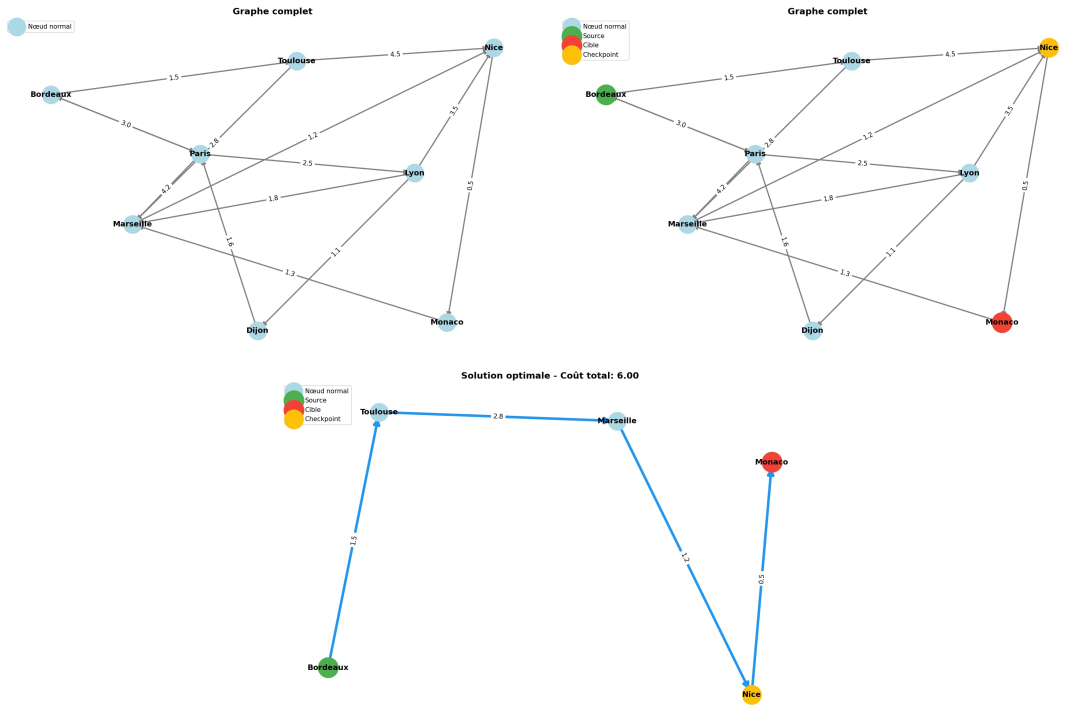

FIGURE 6 – Visualisation du chemin optimal avec checkpoints visités

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

4.5 Formulation Mathématique

4.5.1 Ensembles et Indices

Ensembles

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

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 6 – 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)

4.5.5 Contraintes

C1 - 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.
C2 - 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.
C3 - Capacité des Routes :
Le flux sur une route ne peut dépasser sa capacité :

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

C4 - 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

4.6 Implémentation avec Gurobi

4.6.1 Architecture du Code

Le projet utilise 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

4.6.2 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

4.6.3 Technologies Utilisées

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

4.7 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

4.8 Captures d'Écran de l'Interface

4.8.1 Interface de Saisie

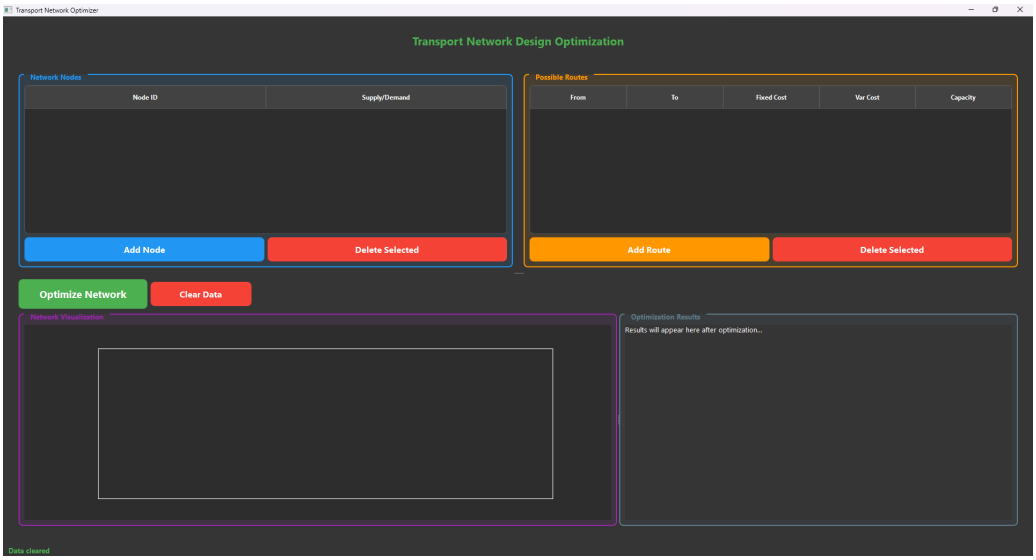

FIGURE 7 – Interface de saisie des entrepôts, clients et routes

4.8.2 Réseau Optimal

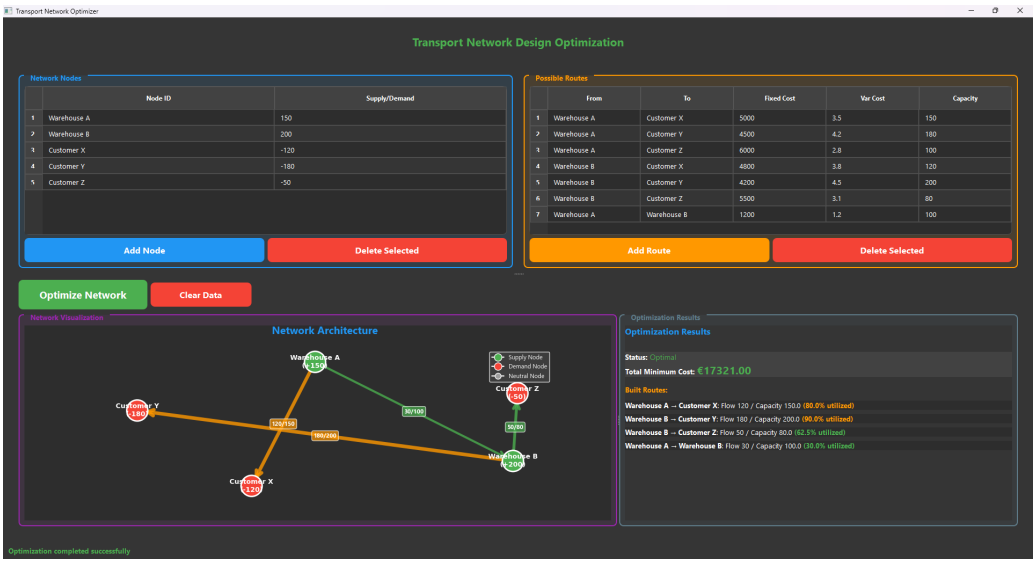

FIGURE 8 – Visualisation du réseau optimal avec flux et routes activées

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

5.5 Formulation Mathématique

5.5.1 Indices et Ensembles

Indices

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

5.5.2 Paramètres

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 7 – Paramètres du modèle - Projet 4

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 :

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

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

5.5.5 Contraintes

C1 - 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.

C2 - 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.

C3 - 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 : Contrainte critique pour le calcul du prix dual (valeur de la capacité).

C4 - 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

C5 - 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).

C6 - 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 du Code

Le projet utilise 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

5.6.2 Étapes de Résolution

1. **Initialisation** : Création du modèle avec variables continues (X_t, A_t, I_t, S_t, C_t) et binaires ($Y_t, Z_t^{inv}, Z_t^{des}$)
2. **Fonction objectif** : Minimisation du coût total actualisé
3. **Contraintes** : Ajout des 6 contraintes (équilibre, capacité, Big-M, mutualité)
4. **Résolution** : Appel à l'algorithme Branch & Bound
5. **Extraction** : Récupération du plan de production optimal

5.6.3 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

5.7 Visualisation et Avantages

5.7.1 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

5.7.2 Garanties du PLM

— **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

5.8 Captures d'Écran de l'Interface

5.8.1 Interface de Planification

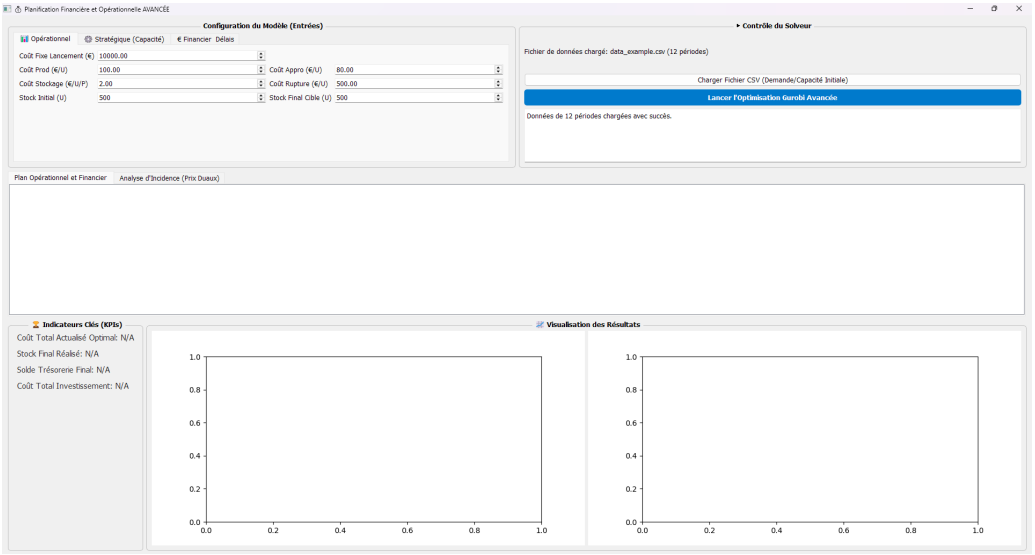

FIGURE 9 – Interface de saisie des périodes, demandes et coûts

5.8.2 Plan de Production Optimal

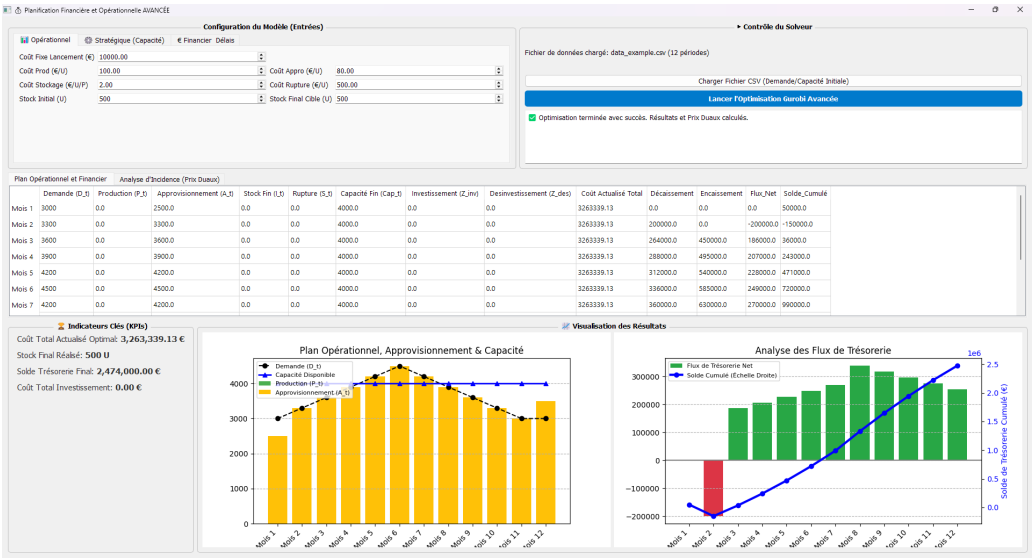

FIGURE 10 – Diagramme de Gantt et plan de production optimal

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 8 – 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 9 – 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)

6.4.5 Contraintes

C1 - 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.

C2 - Respect des Restrictions d'Affectation :

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

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

C3 - Heure de Début Minimale :

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

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

C4 - 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.

C5 - 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

C6 - 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 du Code

Le projet utilise 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

6.5.2 Étapes de Résolution

1. **Initialisation** : Création du modèle 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
5. **Extraction** : Récupération du planning optimal et génération du diagramme de Gantt

6.5.3 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 Captures d'Écran de l'Interface

6.6.1 Interface d'Ordonnancement

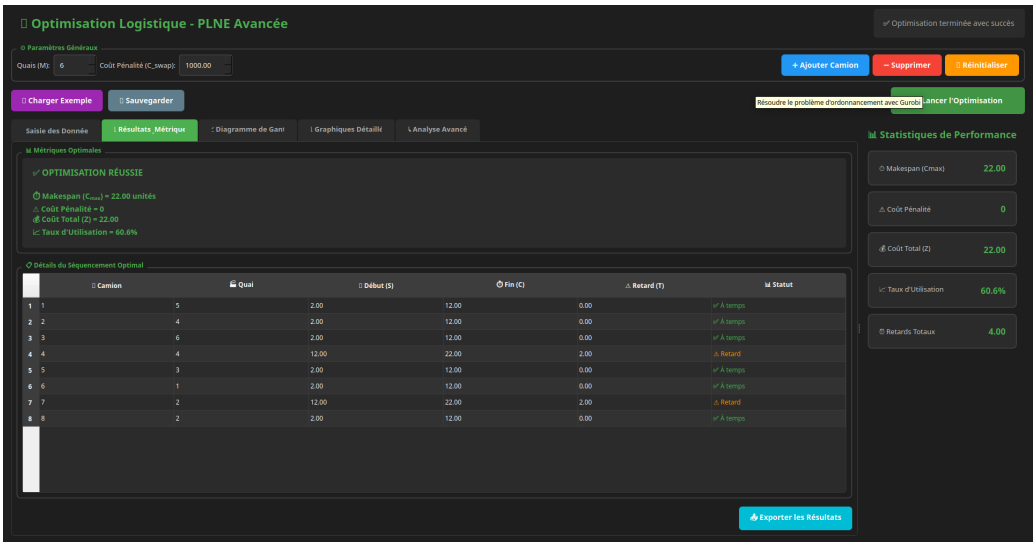

FIGURE 11 – Interface de saisie des camions, quais et contraintes

6.6.2 Planning Optimal

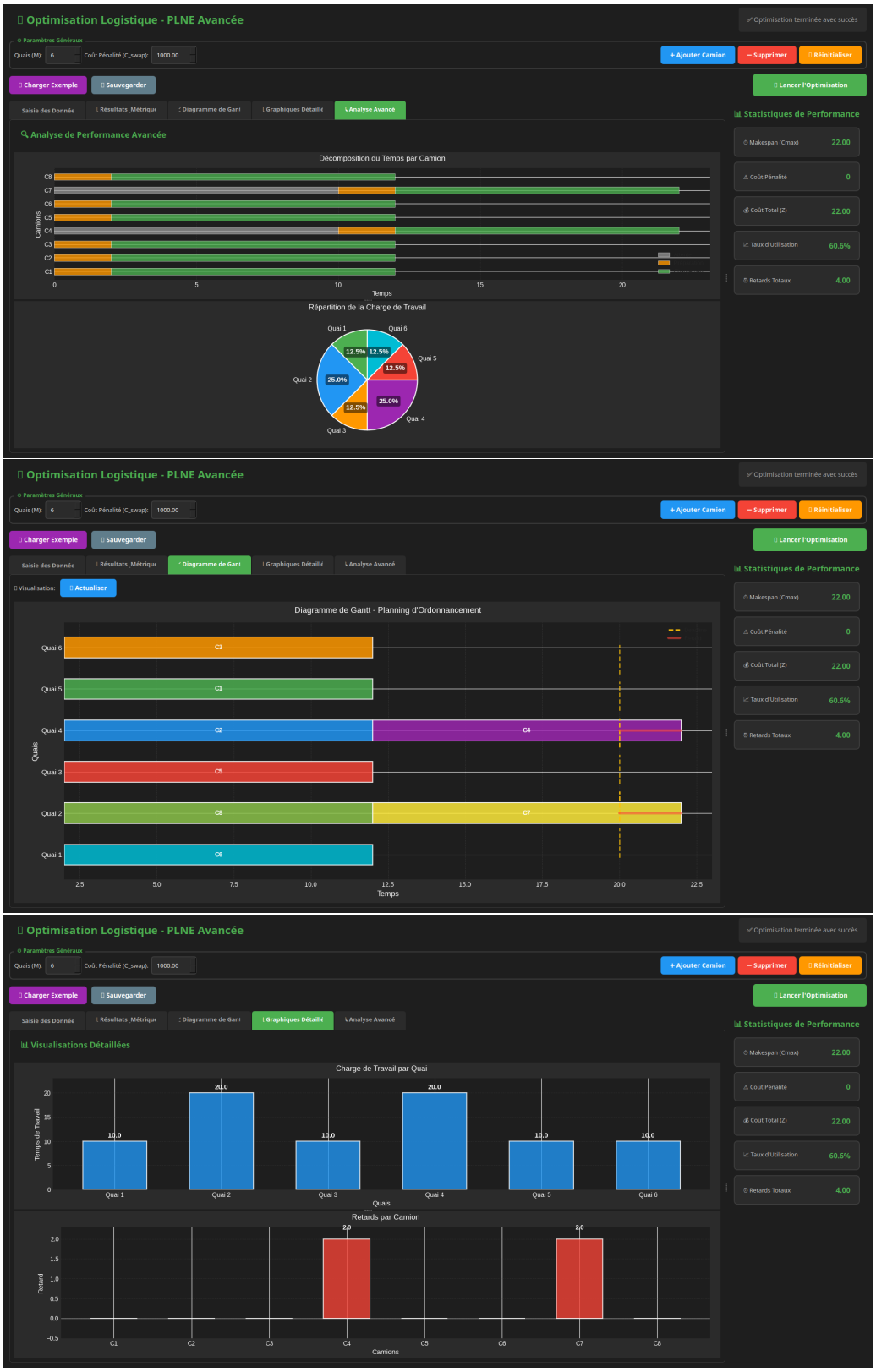

FIGURE 12 – Visualisation du planning optimal avec affectation des camions aux quais

6.7 Avantages du PLNE pour l'Ordonnancement

6.7.1 Comparaison PLNE vs Heuristiques

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

TABLE 10 – PLNE vs 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 11 – 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 12 – 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.

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.