



UNIVERSITÉ SORBONNE PARIS NORD
99 AVENUE JEAN-BAPTISTE CLÉMENT
93430 VILLETANEUSE

SAÉ Optimisation 2025–2026

Conception d'un métro circulaire : le problème Ring-Star

Auteurs :

Papa Birane MBENGUE
Mouhamadou Lamine SOW

Équipe de suivi :

M. Lucas Létocart
M. Pierre Fouilhoux
M. Nabil H. Mustafa

30 janvier 2026

Table des matières

Introduction	1
I NP-difficulté du problème Ring-Star	2
I.1 Cas particulier $p=n$: réduction au TSP	2
I.2 Cas particulier $\alpha=0$: réduction au p-médian	2
I.3 Conséquences	2
II Modélisation mathématique (PLNE)	3
II.1 Variables de décision	3
II.2 Formulation compacte	3
III Algorithmes approchés	4
III.1 Heuristique constructive	4
III.2 Descente stochastique par échanges (SWAP)	4
IV Résultats expérimentaux	5
IV.1 Protocole expérimental	5
IV.2 Analyse qualitative	5
IV.3 Résultats synthétiques	5
IV.4 Courbes coût/temps	8
IV.5 Influence du paramètre p sur la difficulté du PLNE	8

Introduction

Ce rapport étudie le **problème Ring-Star**, une modélisation classique pour concevoir une ligne de métro circulaire (un *anneau*) complétée par des rattachements en *étoile* vers les stations. Sur un ensemble de n sommets d'activité, il s'agit de choisir exactement p stations (dont une station imposée), de construire un cycle passant par ces stations et d'affecter chaque sommet à une station afin de minimiser un coût total combinant *infrastructure* (longueur de l'anneau) et *accessibilité* (distances de rattachement).

Nous montrons la NP-difficulté du problème via des cas particuliers, proposons une formulation PLNE compacte résolue avec **PuLP + CPLEX** (en remplacement de CBC), puis développons une heuristique constructive et une métaheuristique de descente stochastique.

Des expériences sur des instances euclidiennes de TSPLIB (**ulysses16**, **eil51**, **st70**) mettent en évidence (i) le gain d'optimalité apporté par CPLEX, (ii) le bon compromis qualité/temps de la métaheuristique, et (iii) la hausse brutale des temps exacts lorsque p se rapproche de $n/2$.

Définition du problème Ring-Star

Le problème Ring-Star formalise ce compromis. À partir d'un ensemble de n sommets d'activité $V = \{1, \dots, n\}$ et d'une matrice de distances D , on cherche :

- (i) un ensemble $S \subseteq V$ de p stations, avec la station $1 \in S$ imposée ;
- (ii) un cycle (anneau) passant exactement une fois par chaque station de S ;
- (iii) une affectation de chaque sommet $i \in V$ vers une station $a(i) \in S$.

L'objectif est de minimiser un coût total de la forme :

$$\min \quad \alpha \sum_{\{i,j\} \in \text{anneau}} D_{ij} + \sum_{i \in V} D_{i,a(i)}.$$

Le premier terme mesure la longueur de l'anneau (coût d'infrastructure), tandis que le second mesure l'accessibilité via les rattachements en étoile (coût « marche »). La figure 1 illustre le principe.

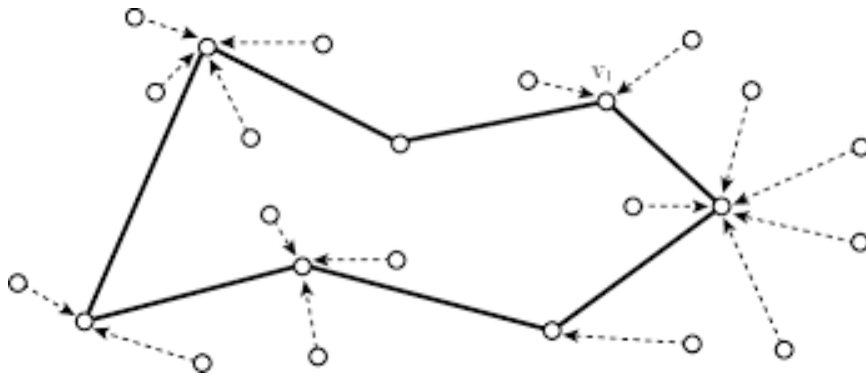


FIGURE 1 – Schéma de principe du Ring-Star : anneau (cycle) sur S et rattachements en étoile.

Instances étudiées

Nos expériences s'appuient sur des instances euclidiennes de TSPLIB : **ulysses16** ($n = 16$), **eil51** ($n = 51$) et **st70** ($n = 70$). La figure 2 montre les nuages de points (données brutes).

La résolution du Ring-Star consiste à choisir p points comme stations, tracer un cycle sur ces stations et affecter les autres points.

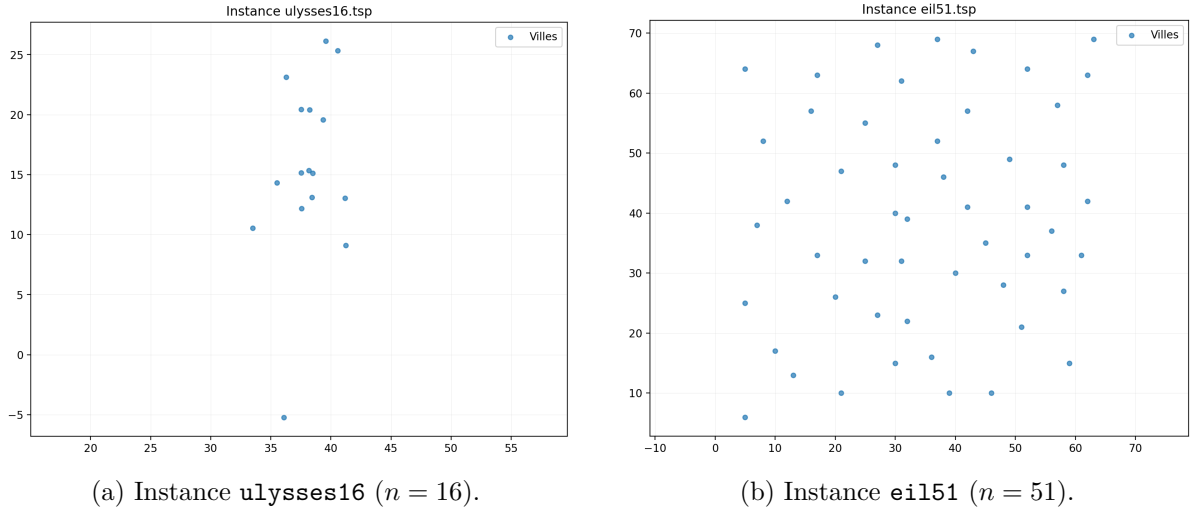


FIGURE 2 – Nuages de points issus de TSPLIB (données d’entrée).

Organisation du rapport

La partie I établit la NP-difficulté. La partie II présente une formulation PLNE compacte. La partie III décrit une heuristique constructive et une métaheuristique d’amélioration. Enfin, la partie IV analyse les résultats expérimentaux et l’influence de p sur la difficulté de la résolution exacte.

I NP-difficulté du problème Ring-Star

Avant de concevoir des algorithmes de résolution, il est essentiel de caractériser la difficulté du problème. Nous montrons la NP-difficulté du Ring-Star en exhibant deux problèmes NP-difficiles comme cas particuliers.

I.1 Cas particulier $p=n$: réduction au TSP

Si $p = n$, chaque pôle est une station. Il n’existe alors aucun rattachement en étoile (coût marche nul) et le problème se réduit à trouver un cycle passant une fois par chaque sommet et minimisant la somme des longueurs : c’est exactement le *problème du voyageur de commerce* (TSP), NP-difficile. Ainsi, Ring-Star généralise le TSP, donc il est NP-difficile.

I.2 Cas particulier $\alpha=0$: réduction au p -médian

Si $\alpha = 0$, le coût de l’anneau devient nul : il ne reste qu’à choisir p stations et à affecter chaque pôle à une station pour minimiser la somme des distances d’affectation. Ce cas particulier correspond au *problème du p -médian*, également NP-difficile. On en déduit à nouveau la NP-difficulté du Ring-Star.

I.3 Conséquences

En pratique, on ne peut pas espérer un algorithme exact polynomial pour toutes les instances. Cela justifie l’usage combiné (i) d’une PLNE pour les petites tailles et (ii) d’heuristiques et

métaheuristiques pour les configurations réalistes.

II Modélisation mathématique (PLNE)

Pour obtenir des solutions optimales sur des instances de taille raisonnable, nous proposons une formulation en **programmation linéaire en nombres entiers (PLNE)** résolue avec PuLP et le solveur **CPLEX** (interface CPLEX_CMD). C'est notre méthode exacte : CPLEX applique un schéma *branch-and-cut* (branch-and-bound, génération de coupes, heuristiques internes, présolve) pour certifier l'optimalité ou un écart.

II.1 Variables de décision

On introduit les variables $y_{ij} \in \{0, 1\}$ (affectation : $y_{ij} = 1$ si le pôle i est affecté à la station j), $x_{ij} \in \{0, 1\}$ (pour $i < j$, arêtes de l'anneau) et $z_{ij} \in [0, p - 1]$ (pour $i \neq j$, flot orienté utilisé pour éliminer les sous-tours et imposer la connexité autour de la station source 1).

II.2 Formulation compacte

$$\min \alpha \sum_{1 \leq i < j \leq n} D_{ij} x_{ij} + \sum_{(i,j) \in V \times V} D_{ij} y_{ij} \quad (1)$$

$$\sum_{i \in V} y_{ii} = p \quad (1)$$

$$\sum_{j \in V} y_{ij} = 1 \quad \forall i \in V \quad (2)$$

$$y_{ij} \leq y_{jj} \quad \forall (i, j) \in V \times V, i \neq j \quad (3)$$

$$\sum_{ij \in \delta(i)} x_{ij} = 2 y_{ii} \quad \forall i \in V \quad (4)$$

$$\sum_{j \in V \setminus \{1\}} z_{1j} = p - 1 \quad (5)$$

$$\sum_{j \in V \setminus \{i\}} z_{ji} = \sum_{j \in V \setminus \{1, i\}} z_{ij} + y_{ii} \quad \forall i \in V \setminus \{1\} \quad (6)$$

$$z_{ij} + z_{ji} \leq (p - 1) x_{ij} \quad \forall i \in V, j \in V \setminus \{1, i\} \quad (7)$$

$$\begin{aligned} x_{ij} &\in \{0, 1\} & \forall ij \in E, \\ y_{ij} &\in \{0, 1\} & \forall (i, j) \in V \times V, \\ z_{ij} &\in [0, p - 1] & \forall (i, j) \in V \times V, i \neq j, \\ y_{11} &= 1, \quad y_{1j} = 0 & \forall j \in V \setminus \{1\}. \end{aligned}$$

- (1) fixe le nombre de stations,
- (2) affecte chaque pôle à une station,
- (3) interdit d'affecter sur un sommet qui n'est pas ouvert,
- (4) impose un degré 2 aux stations pour former un cycle,
- (5) envoie $p - 1$ unités de flot depuis la station imposée 1,
- (6) distribue ce flot en consommant une unité par station atteinte,
- (7) couple flot et arêtes pour éliminer les sous-tours.

III Algorithmes approchés

Compte tenu de la NP-difficulté, les méthodes exactes deviennent rapidement coûteuses. La stratégie pratique est donc : (i) construire vite une solution correcte, (ii) l'améliorer par petites retouches. On décrit ci-dessous l'heuristique gloutonne puis la métaheuristique SWAP qui part de cette base pour la raffiner.

III.1 Heuristique constructive

L'objectif de cette heuristique est de produire rapidement une solution initiale faisable, servant de point de départ aux méthodes d'amélioration. Le principe consiste à sélectionner un ensemble de stations spatialement bien réparties, à relier ces stations par un cycle, puis à affecter chaque pôle à la station la plus proche. Cette approche, qui procède « du centre vers la périphérie », privilégie la rapidité de construction et la faisabilité, sans viser l'optimalité.

L'heuristique gloutonne se décompose en quatre étapes successives :

1. **Initialisation par grille** : une première sélection de stations est obtenue à partir d'une discrétisation de l'espace, afin d'assurer une répartition homogène des stations initiales ;
2. **Complément maximin** : tant que le nombre de stations sélectionnées vérifie $|S| < p$, de nouvelles stations sont ajoutées en maximisant la distance minimale aux stations déjà choisies, renforçant ainsi leur dispersion spatiale ;
3. **Affectation** : chaque pôle est affecté à la station la plus proche, ce qui permet de déterminer les variables d'affectation y ;
4. **Construction du cycle** : un cycle reliant l'ensemble des stations S est construit à l'aide de l'heuristique du plus-proche-voisin pour le problème du voyageur de commerce.

Algorithm 1 Heuristique constructive (gloutonne)

- 1: **Entrée** : points V , entier p , distances D , station imposée 1
 - 2: Construire une grille et extraire un premier ensemble S
 - 3: **while** $|S| < p$ **do**
 - 4: Ajouter au S le pôle maximisant sa distance minimale à S (maximin)
 - 5: **end while**
 - 6: Affecter chaque pôle i à la station $j \in S$ la plus proche (variables y_{ij})
 - 7: Construire un cycle sur S par plus-proche-voisin (variables x_{ij})
 - 8: **Retourner** (S, x, y) et le coût total
-

III.2 Descente stochastique par échanges (SWAP)

Cette métaheuristique s'appuie sur la solution gloutonne initiale qu'elle cherche à améliorer par une série d'échanges locaux aléatoires. À chaque itération, une station est remplacée par un point candidat non station, puis les affectations et le cycle sont recalculés. La nouvelle solution n'est retenue que si elle entraîne une diminution du coût total. La répétition de ces échanges permet de corriger progressivement les choix initiaux, tout en conservant un coût de calcul très faible.

Plus précisément, à partir d'une solution courante, une procédure de descente stochastique explore le voisinage par des opérations de type swap. Une station (à l'exception des stations imposées) est remplacée par un pôle non station, puis les affectations et le cycle correspondants

sont recalculés. La solution obtenue est acceptée uniquement si elle améliore la valeur de la fonction objectif.

Algorithm 2 Métaheuristique de descente stochastique (SWAP)

```

1: Entrée : solution initiale  $(S, x, y)$ , distances  $D$ ,  $\alpha$ , station imposée 1, itérations  $k$ 
2:  $S^* \leftarrow S$  (meilleure solution courante)
3: for  $t = 1$  to  $k$  do
4:   Tirer au hasard une station  $s \in S \setminus \{1\}$  et un pôle  $v \in V \setminus S$ 
5:   Former  $S' \leftarrow (S \setminus \{s\}) \cup \{v\}$ 
6:   Recalculer l'affectation et le cycle sur  $S'$ , obtenir le coût  $C(S')$ 
7:   if  $C(S') < C(S)$  then
8:     Accepter :  $S \leftarrow S'$ 
9:     if  $C(S) < C(S^*)$  then
10:        $S^* \leftarrow S$ 
11:     end if
12:   end if
13: end for
14: Retourner  $S^*$ 

```

IV Résultats expérimentaux

IV.1 Protocole expérimental

Les paramètres sont fixés de manière reproductible : pondération $\alpha = 1$, station imposée : sommet 1 (indice 0 dans le code), graine aléatoire fixée à 0, et nombre d'itérations $k = \max(500, 10n)$. Nous comparons trois approches : heuristique gloutonne, descente stochastique, et PLNE résolue par **PuLP + CPLEX 12.9**. Les instances testées sont **ulysses16** ($n = 16$), **eil51** ($n = 51$) et **st70** ($n = 70$). Les temps reportés sont des temps mur mesurés côté Python.

IV.2 Analyse qualitative

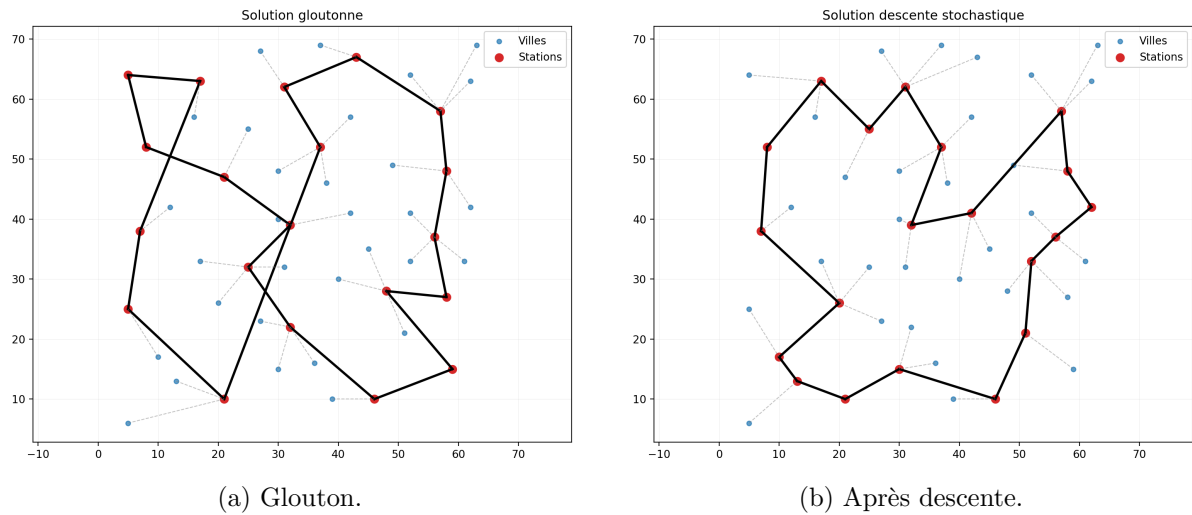
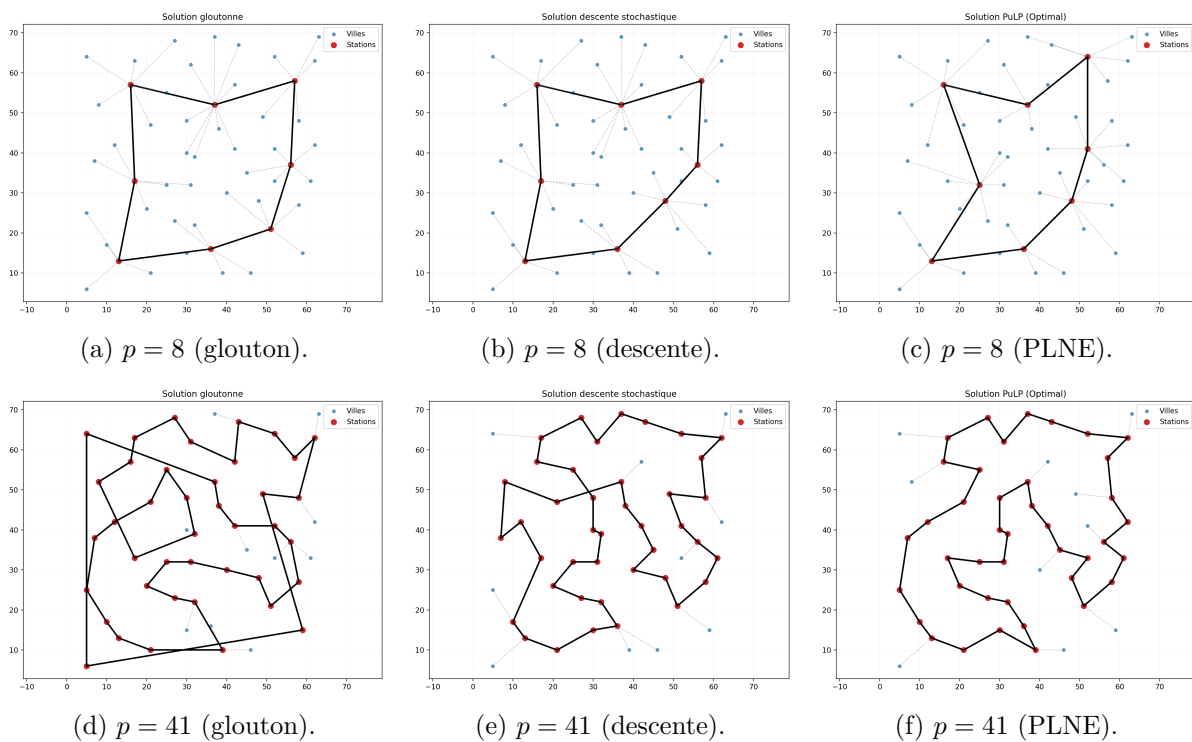
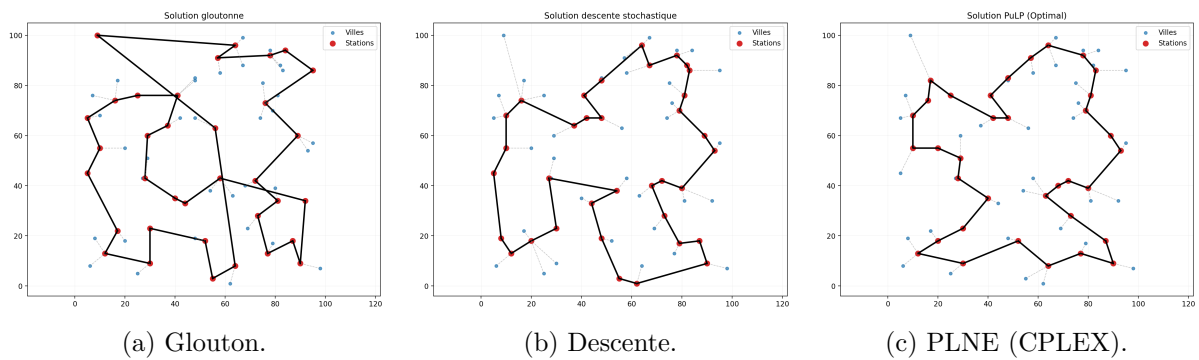
Sur **eil51** avec $p = 20$, la descente stochastique réduit sensiblement le coût et produit un anneau généralement plus régulier (moins de détours) tout en conservant une bonne couverture spatiale (figure 3). CPLEX fournit l'optimum global (voir tableau 2) mais avec un temps nettement plus élevé (plus de 10 minutes sur cette instance).

La figure 4 illustre également deux configurations extrêmes : p faible ($p = 8$) et p élevé ($p = 41$). Quand p est grand, le coût métro domine et la qualité du cycle construit par plus-proche-voisin devient critique : la métaheuristique améliore nettement l'anneau, et la PLNE fournit la référence optimale (en ≈ 2 s pour $p = 41$ avec CPLEX).

Sur **st70** ($n = 70$) avec $p = 35$ (proche de $n/2$), CPLEX atteint l'optimum mais le temps exact devient significatif (≈ 142 s), tandis que la descente stochastique offre un bon compromis coût/temps (figure 5).

IV.3 Résultats synthétiques

Les tableaux 1, 2 et 3 résument coûts (total, métro, marche) et temps. Le *gap* est l'écart relatif au coût optimal PLNE obtenu avec CPLEX.

FIGURE 3 – Comparaison sur `ei151` avec $p = 20$: anneau (cycle) et rattachements en étoile.FIGURE 4 – Comparaisons sur `ei151` pour $p = 8$ (haut) et $p = 41$ (bas).FIGURE 5 – Instance `st70` avec $p = 35$: comparaison glouton / descente stochastique / PLNE.

p	Méthode	Coût total	Métro	Marche	Temps (s)	Gap (%)
5	Glouton	91.964	57.727	34.237	0.000	40.67
5	Descente	65.378	18.248	47.130	0.009	0.00
5	PLNE (opt.)	65.378	18.248	47.130	0.123	0.00
8	Glouton	94.053	72.347	21.706	0.000	47.60
8	Descente	65.015	23.560	41.455	0.011	2.03
8	PLNE (opt.)	63.721	31.363	32.358	0.424	0.00
12	Glouton	104.982	101.653	3.329	0.000	68.55
12	Descente	65.132	38.717	26.415	0.014	4.57
12	PLNE (opt.)	62.286	40.097	22.189	2.048	0.00

TABLE 1 – Résultats sur **ulysses16** ($n = 16$, $\alpha = 1$).

p	Méthode	Coût total	Métro	Marche	Temps (s)	Gap (%)
8	Glouton	594.236	163.678	430.559	0.000	1.25
8	Descente	589.003	160.115	428.887	0.034	0.36
8	PLNE (opt.)	586.893	166.606	420.286	32.919	0.00
20	Glouton	549.603	313.182	236.422	0.001	20.51
20	Descente	482.040	233.842	248.198	0.063	5.70
20	PLNE (opt.)	456.056	209.240	246.815	636.054	0.00
41	Glouton	596.795	535.184	61.611	0.001	47.78
41	Descente	428.565	342.068	86.498	0.109	6.13
41	PLNE (opt.)	403.830	319.788	84.043	2.071	0.00

TABLE 2 – Résultats sur **eil51** ($n = 51$, $\alpha = 1$).

p	Méthode	Coût total	Métro	Marche	Temps (s)	Gap (%)
10	Glouton	1161.818	405.501	756.318	0.001	19.71
10	Descente	992.181	269.764	722.417	0.082	2.23
10	PLNE (opt.)	970.549	266.389	704.160	66.535	0.00
35	Glouton	888.312	658.732	229.579	0.002	36.55
35	Descente	699.111	432.886	266.226	0.192	7.47
35	PLNE (opt.)	650.522	409.668	240.854	141.739	0.00
55	Glouton	818.643	755.906	62.737	0.003	30.98
55	Descente	724.935	609.486	115.449	0.240	15.99
55	PLNE (opt.)	625.014	506.899	118.115	89.698	0.00

TABLE 3 – Résultats sur **st70** ($n = 70$, $\alpha = 1$).

IV.4 Courbes coût/temps

La figure 6 donne une vue d'ensemble (trois instances) : coûts (à gauche) et temps de calcul (à droite, échelle logarithmique). La métaheuristique améliore systématiquement l'heuristique gloutonne en restant sous la seconde, tandis que le PLNE avec CPLEX est optimal mais peut atteindre plusieurs minutes lorsque p est proche de $n/2$.

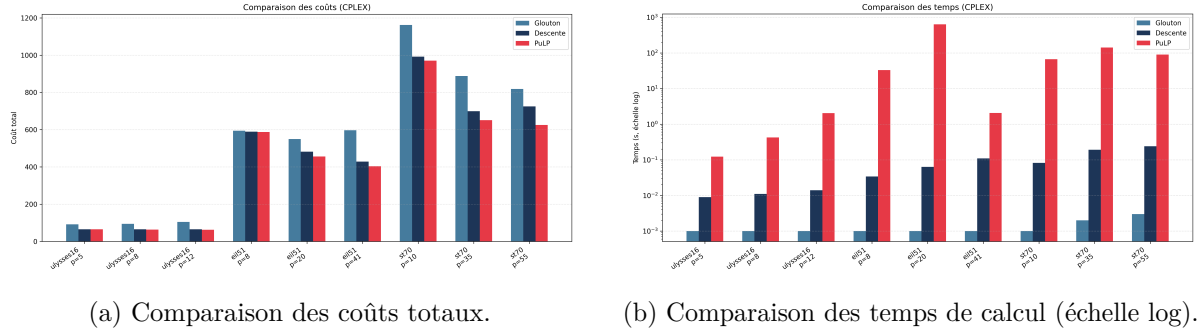


FIGURE 6 – Synthèse des performances par instance et par méthode.

IV.5 Influence du paramètre p sur la difficulté du PLNE

Nos trois bancs de test montrent la même tendance : plus p se rapproche de la moitié des sommets, plus CPLEX peine. Sur la petite instance ulysses16, la courbe reste courte (de 0,12 à 2,05 s) et sert surtout de repère. Sur eil51, on observe un pic à $p = 20$ (636 s) alors que $p = 41$ retombe à 2 s. Sur st70, le maximum survient autour de $p = 35$ (142 s) et redescend ensuite. Autrement dit, la difficulté explose quand la taille de l'anneau est médiane, puis décroît aux extrêmes (anneau très petit ou quasi complet).

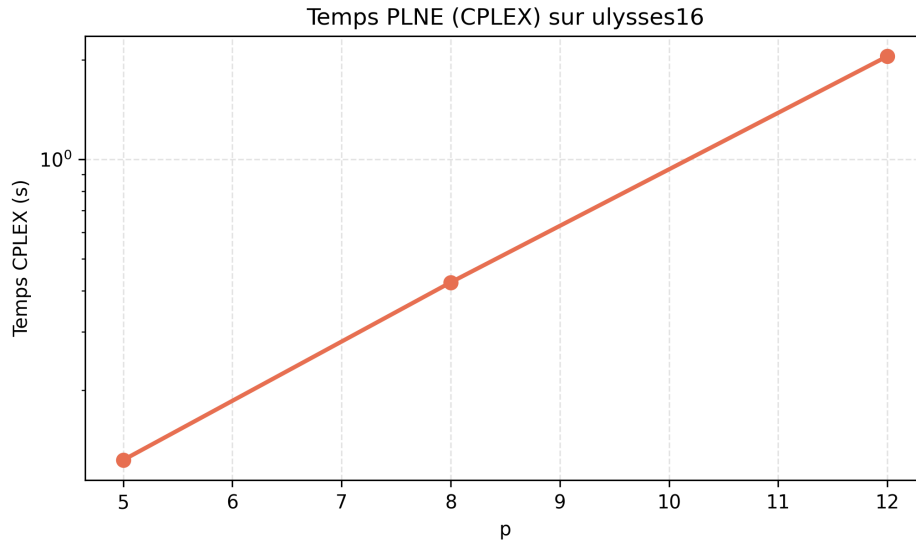


FIGURE 7 – Temps de calcul PLNE (CPLEX) sur ulysses16 en fonction de p .

Sur les instances plus grandes, les temps exacts suivent la même tendance : 636 s pour eil51 avec $p = 20$, entre 66 et 142 s pour st70. D'où l'intérêt de limiter CPLEX aux petites tailles ou à des valeurs de p éloignées de $n/2$, et de s'appuyer sur les heuristiques dans les autres cas. La figure 8 juxtapose les courbes $p \mapsto$ temps CPLEX pour eil51 et st70.

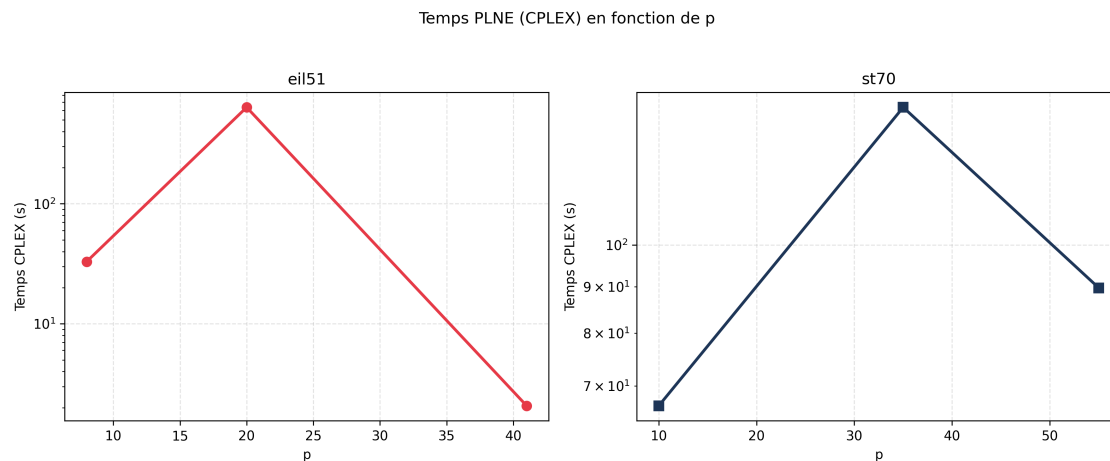


FIGURE 8 – Temps de résolution PLNE (CPLEX) en fonction de p sur **eil51** (gauche) et **st70** (droite), échelle logarithmique.

Conclusion générale

Ce travail a permis d'aborder le problème Ring-Star à la fois sous l'angle de la modélisation et de la résolution pratique. Les résultats confirment que, malgré une formulation exacte efficace sur des instances de taille modérée, la complexité du problème limite rapidement l'usage exclusif des méthodes exactes lorsque la taille augmente.

Les approches heuristiques proposées constituent alors une alternative pertinente : elles permettent d'obtenir très rapidement des solutions faisables, puis de les améliorer de manière significative avec un coût de calcul négligeable. Les expérimentations montrent ainsi qu'un compromis efficace peut être trouvé entre qualité de solution et temps de calcul.

En pratique, une stratégie hybride apparaît la plus adaptée : les méthodes exactes peuvent être mobilisées pour des instances de petite taille ou à des fins de validation, tandis que les heuristiques offrent une solution robuste et scalable pour les cas plus réalistes. Cette complémentarité illustre l'intérêt de combiner rigueur théorique et pragmatisme algorithmique pour traiter des problèmes d'optimisation complexes.