



Mémoire de fin d'étude

Pour l'obtention du diplôme de master

Filière : Génie industriel

Spécialité : Management industriel et logistique

Présenté par : Rym TALEB

Thème

Les grands modèles de langage (LLM) comme assistants guidés

pour la résolution de problèmes d'optimisation combinatoire

Soutenu publiquement, le 18 / 06 / 2026 , devant le jury composé de:

Mohammed Benekfrouf	Maître de conférences A	ESSA Tlemcen	Président
Fouad Maliki	Maître de conférences A	ESSA Tlemcen	Directeur de mémoire
Nadjib Brahimi	Professeur	IMT Atlantique / Rennes SB	Co-directeur de mémoire
Mehdi Souier	Professeur	Université de Tlemcen	Examineur 1
Abdelghani Bekrar	Maître de conférences	UPHF	Examineur 2
Sadeque Hamdan	Maître de conférences	Bangor University / Bangor BS	Invité

Année universitaire :2025/2026

بِسْمِ اللَّهِ الرَّحْمَنِ الرَّحِيمِ

Dédicace

Je dédie ce travail à mes chers parents, pour leur amour, leur soutien inconditionnel, leur patience et leurs encouragements constants tout au long de mon parcours. Depuis mon enfance jusqu'à l'aboutissement de cette étape importante de ma formation, leur présence, leurs sacrifices et leur confiance ont été une source essentielle de force et de motivation.

Je dédie également ce mémoire à mes sœurs et à mes beaux-frères, pour leur soutien, leur bienveillance et leurs encouragements. Leur présence à mes côtés a toujours représenté pour moi un appui précieux dans les moments de doute comme dans les moments de réussite.

À mes petits neveux Ahmed et Mohamed Mounir, ainsi qu'à ma petite nièce Asma, dont l'innocence, la joie et l'affection apportent beaucoup de bonheur à ma vie. Leur présence me rappelle l'importance de persévérer et de construire un avenir dont ma famille pourra être fière.

Je souhaite également dédier ce mémoire à Monsieur Fouad Maliki, mon encadrant académique et responsable de formation, qui a profondément marqué mon parcours. Après des années de classes préparatoires particulièrement difficiles, il a su me faire découvrir et aimer le Génie Industriel. Par sa pédagogie, son accompagnement, sa bienveillance et son exigence, il a été pour moi bien plus qu'un enseignant : l'un des meilleurs professeurs que j'ai eu la chance de rencontrer tout au long de mon parcours.

Je souhaite également adresser une pensée particulière à Monsieur Nadjib Brahim, mon encadrant de stage, qui m'a fait confiance et m'a donné l'opportunité de rejoindre cette expérience de recherche. Cette confiance a représenté pour moi une véritable source de motivation et m'a permis de me sentir pleinement impliquée dans ce travail. Son accompagnement, sa disponibilité et ses encouragements ont contribué à rendre cette expérience à la fois enrichissante, formatrice et marquante dans mon parcours.

J'adresse également une pensée sincère à mes amies Kawther, Hind, Sara et Ghizlene, avec lesquelles j'ai partagé cette formation en Génie Industriel. Je les remercie pour leur amitié, leur soutien, les moments partagés et les souvenirs précieux construits tout au long de ce parcours.

Enfin, je dédie ce mémoire à toutes les personnes qui ont cru en moi, qui m'ont encouragée et qui m'ont accompagnée, de près ou de loin, jusqu'à l'obtention de ce diplôme.

Rym

Remerciements

Au terme de ce travail, je tiens à exprimer ma profonde gratitude à toutes les personnes qui ont contribué, de près ou de loin, à la réalisation de ce mémoire.

Je souhaite tout d'abord remercier les membres du jury pour l'honneur qu'ils me font en acceptant d'évaluer ce travail. Je leur suis reconnaissante pour le temps consacré à la lecture de ce mémoire, ainsi que pour leurs remarques et recommandations qui contribueront à enrichir ma réflexion.

J'adresse mes sincères remerciements à mon encadrant académique, Monsieur Fouad Maliki, pour son suivi, ses remarques constructives, son soutien moral et ses encouragements durant la réalisation de ce mémoire. Ses conseils ont contribué à améliorer la cohérence scientifique et méthodologique de ce travail, et son accompagnement m'a été précieux tout au long de cette période.

Je remercie également Monsieur Nadjib Brahimi, mon encadrant de stage, pour son accompagnement, sa disponibilité, ses conseils et la confiance qu'il m'a accordée tout au long de ce travail. Je le remercie particulièrement de m'avoir donné l'opportunité de rejoindre cette expérience de recherche et de m'avoir guidée avec bienveillance et exigence.

J'adresse également ma profonde reconnaissance et mes sincères remerciements à Monsieur Sadeque Hamdan pour son aide, ses échanges et son appui tout au long de ce travail. Son accompagnement, ses remarques et sa disponibilité ont été précieux pour renforcer la qualité de ma démarche et consolider les résultats présentés dans ce mémoire.

Je tiens également à remercier l'IMT Atlantique, le laboratoire LS2N ainsi que l'équipe Modelis pour leur accueil et pour l'environnement de travail stimulant dont j'ai pu bénéficier durant mon stage. Cette expérience m'a permis de découvrir davantage le monde de la recherche.

Mes remerciements s'adressent aussi à l'École Supérieure en Sciences Appliquées de Tlemcen, ainsi qu'à l'ensemble des enseignants qui ont contribué à ma formation en Génie Industriel. Les connaissances acquises tout au long de mon parcours ont constitué une base essentielle pour mener à bien ce travail.

Enfin, j'adresse mes remerciements les plus profonds à ma famille pour son soutien constant, sa patience, ses encouragements et sa confiance. Je remercie également mes amis et toutes les personnes qui m'ont accompagnée, soutenue ou encouragée durant cette période.

À toutes ces personnes, j'adresse mes plus sincères remerciements.

ملخص

تتناول هذه المذكرة تقييم النماذج اللغوية الكبيرة باعتبارها أدوات مساعدة في حل مسائل التحسين التوافقي. ففي الأنظمة الصناعية واللوجستية، تتطلب القرارات المتعلقة باختيار المواقع، والتغطية، وتخطيط الإنتاج، وتنظيم المسارات، استعمال طرائق دقيقة قادرة على إنتاج حلول قابلة للتنفيذ، وذات جودة جيدة، وقابلة للتحقق. وقد أتاح ظهور هذه النماذج آفاقاً جديدة في مجال المساعدة على النمذجة، وكتابة البرامج، واقتراح طرائق الحل. غير أن موثوقيتها تبقى مسألة أساسية، إذ يمكن أن تكون الإجابة مصاغة بشكل جيد، لكنها تحتوي على أخطاء في الاستدلال، أو الحساب، أو التحقق من قابلية الحل للتنفيذ.

يهدف هذا العمل إلى تحليل مدى قدرة النماذج اللغوية الكبيرة على المساعدة في حل مسائل التحسين التوافقي، مع تحديد حدودها. وتعتمد الدراسة على منهجية تجريبية مكونة من مرحلتين. في المرحلة الأولى، يتم تقييم النماذج بطريقة شبه مستقلة، بهدف ملاحظة قدرتها على فهم المسائل، واقتراح طريقة للحل، وإنتاج حل مناسب. أما المرحلة الثانية، فتدخل توجيهها قائماً على طرائق تقريبية، حيث يطلب من النماذج تطبيق طريقة محددة والتحقق من النتائج المتحصل عليها.

تركز التجارب أساساً على ثلاث مسائل كلاسيكية في التحسين التوافقي، وهي مسألة اختيار المراكز، ومسألة تغطية المجموعات، ومسألة تخطيط الإنتاج غير المقيد. وقد تم تقييم إجابات النماذج اعتماداً على عدة مؤشرات، من بينها قابلية الحل للتنفيذ، وقيمة دالة الهدف، والفجوة مقارنة بحل مرجعي، والموثوقية، وصحة التنفيذ البرمجي، ومدى التحسن مقارنة بالمرحلة الأولى. تظهر النتائج أن النماذج اللغوية الكبيرة يمكن أن تستفيد أحياناً من التوجيه القائم على طرائق تقريبية، خاصة من أجل تحسين قابلية الحل للتنفيذ أو تنظيم مسار الحل. ومع ذلك، يبقى أداؤها غير مستقر حسب النموذج والمسألة المدروسة. وعليه، يؤكد هذا العمل أن هذه النماذج لا يمكن اعتبارها برامج حل مستقلة وموثوقة، لكنها يمكن أن تؤدي دور مساعدين موجّهين، بشرط أن يتم التحقق من نتائجها بشكل منهجي باستعمال طرائق خارجية.

Abstract

This thesis focuses on the evaluation of Large Language Models (LLMs) as assistance tools for solving combinatorial optimization problems. In industrial and logistics systems, decisions related to facility location, set covering, production planning, or routing require rigorous methods capable of producing feasible, efficient, and verifiable solutions. The emergence of LLMs opens new perspectives for modeling assistance, code generation, and the proposal of solution methods. However, their reliability remains a central issue, since a well-structured answer may still contain reasoning, calculation, or feasibility errors.

The objective of this work is to analyze to what extent LLMs can assist in solving combinatorial optimization problems, while identifying their limitations. The study is based on a two-phase experimental approach. The first phase evaluates the models in a relatively autonomous setting, in order to observe their ability to understand the problems, propose a solution method, and generate a solution. The second phase introduces heuristic guidance, where the models are required to apply an imposed method and verify the obtained results.

The experiments mainly focus on three classical problems: the p -median problem, the *Set Covering Problem*, and the *Uncapacitated Lot Sizing Problem*. The responses generated by the LLMs are evaluated using several indicators, including feasibility, objective value, gap with respect to a reference value, reliability, correctness of implementation, and improvement compared to the initial phase.

The results show that LLMs can sometimes benefit from heuristic guidance, particularly by improving feasibility or structuring the resolution process. However, their performance remains unstable depending on the model and the problem studied. This work therefore confirms that LLMs cannot be considered reliable autonomous solvers, but that they can play the role of guided assistants, provided that their results are systematically verified using external methods.

Résumé

Ce mémoire porte sur l'évaluation des grands modèles de langage, ou *Large Language Models* (LLM), comme outils d'assistance à la résolution de problèmes d'optimisation combinatoire. Dans les systèmes industriels et logistiques, les décisions liées à la localisation, à la couverture, à la planification de production ou au routage nécessitent des méthodes rigoureuses permettant de produire des solutions faisables, performantes et vérifiables. L'émergence des LLM ouvre de nouvelles perspectives pour l'aide à la modélisation, la génération de code et la proposition de méthodes de résolution. Cependant, leur fiabilité reste une question centrale, car une réponse bien formulée peut contenir des erreurs de raisonnement, de calcul ou de faisabilité.

L'objectif de ce travail est d'analyser dans quelle mesure les LLM peuvent assister la résolution de problèmes d'optimisation combinatoire, tout en identifiant leurs limites. L'étude repose sur une démarche expérimentale en deux phases. La première phase évalue les modèles de manière relativement autonome, afin d'observer leur capacité à comprendre les problèmes, proposer une méthode et produire une solution. La deuxième phase introduit un guidage heuristique, dans lequel les modèles doivent appliquer une méthode imposée et vérifier les résultats obtenus.

Les expérimentations portent principalement sur trois problèmes classiques : le problème du p -médian, le *Set Covering Problem* et l'*Uncapacitated Lot Sizing Problem*. Les réponses des LLM sont évaluées à partir de plusieurs indicateurs, notamment la faisabilité, la valeur objective, le gap par rapport à une référence, la fiabilité, la correction de l'implémentation et l'amélioration par rapport à la phase initiale.

Les résultats montrent que les LLM peuvent parfois bénéficier d'un guidage heuristique, notamment pour améliorer la faisabilité ou structurer la résolution. Toutefois, leurs performances restent instables selon les modèles et les problèmes étudiés. Ce travail confirme donc que les LLM ne peuvent pas être considérés comme des solveurs autonomes fiables, mais qu'ils peuvent jouer un rôle d'assistants guidés, à condition que leurs résultats soient systématiquement vérifiés par des méthodes externes.

Table des matières

Dédicace	2
Remerciements	3
Résumé en arabe	4
Abstract	5
Résumé	6
Liste des figures	11
Liste des tableaux	12
Liste des abréviations	13
Introduction générale	15
Introduction générale	15
1 Optimisation combinatoire : modélisation, méthodes de résolution et hybridation IA-RO	18
1.1 Introduction	18
1.2 Introduction à l'optimisation	19
1.2.1 Définition générale	19
1.2.2 Minimisation et maximisation	19
1.2.3 Rôle des contraintes	19
1.2.4 Solution faisable vs solution optimale	20
1.3 Typologie des problèmes d'optimisation	20
1.3.1 Optimisation continue vs discrète	20
1.3.2 Optimisation linéaire vs non linéaire	20
1.3.3 Déterministe vs stochastique et robuste	20
1.3.4 Mono-objectif vs multi-objectifs	21
1.4 Optimisation combinatoire : définitions et complexité	21
1.4.1 Définition formelle	21
1.4.2 Explosion combinatoire	22
1.4.3 Théorie de la complexité : NP-difficulté	22

1.5	Modélisation mathématique : bonnes pratiques de formulation	23
1.5.1	Variables, contraintes et interprétation opérationnelle	23
1.5.2	Renforcement de formulation	23
1.5.3	Gestion des symétries	23
1.5.4	Prétraitement et réduction d'instance	23
1.6	Méthodes exactes en optimisation combinatoire	24
1.6.1	Programmation linéaire en nombres entiers	24
1.6.2	Relaxation linéaire et bornes	24
1.6.3	Branch-and-Bound	25
1.6.4	Branch-and-Cut	25
1.6.5	Génération de colonnes	25
1.6.6	Avantages et limites des méthodes exactes	25
1.7	Heuristiques et recherche locale	26
1.7.1	Heuristiques constructives	26
1.7.2	Recherche locale	26
1.7.3	Méthodes à population	27
1.7.4	Réparation de faisabilité	27
1.8	Métaheuristiques	27
1.8.1	Définition et objectifs	27
1.8.2	Recuit simulé	27
1.8.3	Recherche tabou	27
1.8.4	Algorithmes génétiques	28
1.8.5	GRASP, ILS et VNS	28
1.8.6	ALNS : Adaptive Large Neighborhood Search	28
1.8.7	LLM et métaheuristiques	29
1.9	Les trois problèmes étudiés : modélisations mathématiques	29
1.9.1	Problème du p -médian	29
1.9.2	Set Covering Problem	30
1.9.3	Uncapacitated Lot Sizing	31
1.10	Hybridation IA–Recherche Opérationnelle	33
1.10.1	Pourquoi hybrider ?	33
1.10.2	Formes d'hybridation IA–RO	33
1.10.3	Place des LLM dans l'hybridation	34
1.10.4	LLM + solveur : intégration et ancrage	34
1.11	Conclusion	35
2	Présentation de l'organisme d'accueil et du contexte du stage	36
2.1	Introduction	36
2.2	Présentation générale d'IMT Atlantique	37
2.2.1	Identité et positionnement	37
2.2.2	Missions principales	37
2.2.3	Lien avec le sujet du stage	38
2.3	Le campus de Nantes	38
2.3.1	Localisation et environnement scientifique	38
2.3.2	Un cadre adapté au stage	38
2.4	Présentation du LS2N	39
2.4.1	Identité du laboratoire	39
2.4.2	Organisation générale	39

2.4.3	Lien avec les sciences de la décision	39
2.5	Présentation de l'équipe MODELIS	40
2.5.1	Identité et rattachement	40
2.5.2	Axes scientifiques	40
2.6	Organisation de la recherche en France et comparaison avec le contexte algérien	40
2.6.1	Structure générale de la recherche en France	40
2.6.2	Comparaison avec le contexte algérien	41
2.7	Contexte général du stage	42
2.7.1	Sujet du stage	42
2.7.2	Motivation scientifique	42
2.7.3	Motivation industrielle	42
2.7.4	Positionnement dans le mémoire	43
2.8	Apports du stage	43
2.8.1	Apports scientifiques	43
2.8.2	Apports méthodologiques	43
2.8.3	Apports pour le génie industriel	43
2.8.4	Apports personnels et professionnels	44
2.8.5	Immersion dans un environnement scientifique	44
2.9	Difficultés rencontrées	45
2.9.1	Complexité du sujet	45
2.9.2	Fiabilité des réponses des LLM	45
2.9.3	Traçabilité expérimentale	45
2.10	Lien entre le stage et la formation	45
2.11	Conclusion	46
3	Evaluation guidée des LLM sur des problèmes d'optimisation	47
3.1	Introduction	47
3.2	Principe général de la phase 2	47
3.3	Sélection des modèles évalués	48
3.4	Construction du guidage heuristique	48
3.5	Configuration des prompts	49
3.5.1	Structure générale des prompts	50
3.5.2	Contraintes imposées aux LLM	50
3.5.3	Heuristiques guidées utilisées	51
3.5.4	Format de sortie demandé	54
3.5.5	Informations de performance demandées	55
3.6	Indicateurs d'évaluation de la phase 2	55
3.7	Objectif de comparaison avec la phase 1	56
3.8	Résultats obtenus sur le problème du p -médián	57
3.8.1	Analyse globale par modèle	58
3.8.2	Analyse de la faisabilité	58
3.8.3	Analyse de la qualité objective et du gap	59
3.8.4	Analyse de la fiabilité	60
3.8.5	Analyse de la correction de l'implémentation	60
3.8.6	Comparaison avec la phase 1 à travers l'indicateur Improvement / Phase 1	61
3.8.7	Synthèse des résultats sur le p -médián	62

3.9	Résultats obtenus sur le <i>Set Covering Problem</i>	62
3.9.1	Analyse globale par modèle	63
3.9.2	Analyse de la faisabilité	64
3.9.3	Analyse de la qualité objective et du gap	64
3.9.4	Analyse de la fiabilité	65
3.9.5	Analyse de la correction de l'implémentation	65
3.9.6	Comparaison avec la phase 1 à travers l'indicateur Improvement / Phase 1	66
3.9.7	Synthèse des résultats sur le <i>Set Covering Problem</i>	66
3.10	Résultats obtenus sur l' <i>Uncapacitated Lot Sizing Problem</i>	67
3.10.1	Analyse globale par modèle	68
3.10.2	Analyse de la faisabilité	69
3.10.3	Analyse de la qualité objective et du gap	69
3.10.4	Analyse de la fiabilité	70
3.10.5	Analyse de la correction de l'implémentation	70
3.10.6	Comparaison avec la phase 1 à travers l'indicateur Improvement / Phase 1	71
3.10.7	Synthèse des résultats sur le Lot Sizing	72
3.11	Conclusion	72
	Conclusion générale	74
	Conclusion	74
	Annexes	76
A	Prompts guidés utilisés dans la phase 2	76
A.1	Prompt guidé du problème <i>p</i> -median	76
A.1.1	Prompt guidé	76
A.2	Prompt guidé du <i>Set Covering Problem</i>	78
A.2.1	Prompt guidé	79
A.3	Prompt guidé de l' <i>Uncapacitated Lot Sizing Problem</i>	80
A.3.1	Prompt guidé	80
	Bibliographie	84

Liste des figures

1.1	Typologie générale des problèmes d’optimisation	21
1.2	Classification des principales méthodes de résolution en optimisation combinatoire	24
1.3	Boucle d’intégration entre LLM et solveur pour l’ancrage des solutions	34

Liste des tableaux

1.1	Distinction entre solution faisable et solution optimale	20
1.2	Classes de complexité et implications pratiques	22
1.3	Avantages et limites des méthodes exactes	25
1.4	Synthèse des principales métaheuristiques	28
1.5	Comparaison des trois problèmes étudiés	32
1.6	Formes d'hybridation IA-RO et exemples	33
2.1	Synthèse de l'organisation du LS2N	39
3.1	Heuristiques guidées utilisées dans la phase 2	51
3.2	Indicateurs d'évaluation utilisés dans la phase 2	55
3.3	Bilan global des résultats de la phase 2 sur le problème du p-médian	57
3.4	Synthèse des performances par LLM en phase 2 sur le p-médian	58
3.5	Comparaison avec la phase 1 selon l'indicateur Improvement / Phase 1	61
3.6	Bilan global des résultats de la phase 2 sur le Set Covering Problem	62
3.7	Synthèse des performances par LLM en phase 2 sur le Set Covering Problem	63
3.8	Comparaison avec la phase 1 selon l'indicateur Improvement / Phase 1 pour le Set Covering Problem	66
3.9	Bilan global des résultats de la phase 2 sur le Lot Sizing	67
3.10	Synthèse des performances par LLM en phase 2 sur le Lot Sizing	68
3.11	Comparaison avec la phase 1 selon l'indicateur Improvement / Phase 1 pour le Lot Sizing	71

Liste des abréviations

ESSAT — École Supérieure en Sciences Appliquées de Tlemcen

Chapitre 1

IA — Intelligence Artificielle

RO — Recherche Opérationnelle

IA-RO — Hybridation entre Intelligence Artificielle et Recherche Opérationnelle

LLM — Large Language Model, grand modèle de langage

GenAI — Generative Artificial Intelligence, intelligence artificielle générative

LP — Linear Programming, programmation linéaire

NLP — Nonlinear Programming, programmation non linéaire

ILP — Integer Linear Programming, programmation linéaire en nombres entiers

MILP — Mixed Integer Linear Programming, programmation linéaire mixte en nombres entiers

P — Classe des problèmes résolubles en temps polynomial

NP — Classe des problèmes vérifiables en temps polynomial

SAT — Satisfiability Problem, problème de satisfiabilité booléenne

TSP — Traveling Salesman Problem, problème du voyageur de commerce

UPMP — Uncapacitated P-Median Problem, problème de la p -médiane non capacisée

SCP — Set Covering Problem, problème de couverture par ensembles

ULS — Uncapacitated Lot Sizing Problem, problème de dimensionnement de lots non capacitaire

OR-Library — Operations Research Library, bibliothèque d'instances de référence en recherche opérationnelle

ML — Machine Learning, apprentissage automatique

RL — Reinforcement Learning, apprentissage par renforcement

GRASP — Greedy Randomized Adaptive Search Procedure

ILS — Iterated Local Search

VNS — Variable Neighborhood Search

ALNS — Adaptive Large Neighborhood Search

VRP — Vehicle Routing Problem, problème de tournées de véhicules
VRPDO — Vehicle Routing Problem with Delivery Options, problème de tournées de véhicules avec options de livraison
CPLEX — Solveur d’optimisation mathématique développé par IBM
Gurobi — Solveur d’optimisation mathématique
OR-Tools — Suite d’outils d’optimisation développée par Google
SIRL — Solver-Informed Reinforcement Learning

Chapitre 2

IMT — Institut Mines-Télécom
IMT Atlantique — École d’ingénieurs française relevant de l’Institut Mines-Télécom
LS2N — Laboratoire des Sciences du Numérique de Nantes
UMR — Unité Mixte de Recherche
CNRS — Centre National de la Recherche Scientifique
Inria — Institut National de Recherche en Sciences et Technologies du Numérique
MODELIS — Modélisation, Optimisation et Décision pour la Logistique, l’Industrie et les Services
DAPI — Département Automatique, Productique et Informatique
LLM — Large Language Model, grand modèle de langage
IA — Intelligence Artificielle
RO — Recherche Opérationnelle
ROADEF — Société Française de Recherche Opérationnelle et d’Aide à la Décision

Chapitre 3

LLM — Large Language Model, grand modèle de langage
SCP — Set Covering Problem, problème de couverture par ensembles
ULS — Uncapacitated Lot Sizing Problem, problème de dimensionnement de lots non capacitaire
GAP — Écart relatif entre la solution obtenue et la solution optimale ou de référence
MILP — Mixed Integer Linear Programming, programmation linéaire mixte en nombres entiers
CPU — Central Processing Unit, unité centrale de traitement ; utilisé ici pour désigner le temps de calcul ou d’exécution
OR-Library — Operations Research Library, bibliothèque d’instances de référence en recherche opérationnelle
ADD — Méthode d’ajout progressif utilisée dans l’heuristique *Greedy ADD*
S.A. — Simulated Annealing, recuit simulé

Introduction

Les systèmes industriels et logistiques actuels sont confrontés à des décisions de plus en plus complexes. La planification de la production, la localisation d’installations, la gestion des stocks, la couverture de services ou encore l’organisation des flux nécessitent de prendre des décisions sous contraintes, tout en cherchant à minimiser les coûts, les distances, les délais ou les ressources mobilisées. Dans ce contexte, l’optimisation combinatoire occupe une place centrale en génie industriel et en recherche opérationnelle, car elle permet de modéliser rigoureusement ces décisions et de rechercher des solutions performantes.

Cependant, les problèmes d’optimisation combinatoire sont souvent difficiles à résoudre. Lorsque le nombre de décisions augmente, l’espace des solutions possibles devient très vaste, ce qui rend l’énumération exhaustive irréaliste. Les méthodes exactes, telles que la programmation linéaire en nombres entiers, le *Branch-and-Bound* ou le *Branch-and-Cut*, permettent d’obtenir des garanties d’optimalité, mais peuvent devenir coûteuses sur de grandes instances. Les heuristiques et métaheuristiques constituent alors des alternatives utiles, car elles permettent de produire rapidement des solutions faisables et de bonne qualité, même sans garantie systématique d’optimalité.

Parallèlement, les progrès récents de l’intelligence artificielle générative ont conduit à une utilisation croissante des grands modèles de langage, ou *Large Language Models* (LLM), dans des tâches variées : rédaction, programmation, explication de concepts, génération de code, aide à la modélisation et résolution de problèmes. Leur capacité à produire des raisonnements structurés et des réponses détaillées soulève une question importante pour la recherche opérationnelle : ces modèles peuvent-ils assister efficacement la résolution de problèmes d’optimisation combinatoire ?

Cette question est particulièrement importante, car une réponse produite par un LLM peut être bien formulée sans être correcte. En optimisation, la qualité d’une solution ne peut pas être jugée uniquement sur la clarté de l’explication. Elle doit être vérifiée à partir de critères objectifs : respect des contraintes, faisabilité, valeur de la fonction objectif, écart à une solution optimale ou de référence, et cohérence entre la solution annoncée et la solution réellement obtenue. Ainsi, l’utilisation des LLM dans un contexte d’aide à la décision nécessite une évaluation rigoureuse.

Ce mémoire s’inscrit dans cette problématique. Il vise à étudier la capacité des LLM à traiter des problèmes d’optimisation combinatoire issus de la logistique, de la production et de la recherche opérationnelle. L’objectif n’est pas de considérer les LLM comme des solveurs autonomes, mais d’analyser dans quelle mesure ils peuvent jouer le rôle d’assistants capables de comprendre un problème, proposer une méthode, générer une solution, appliquer une heuristique et produire des résultats vérifiables.

Le travail réalisé s’inscrit dans le cadre d’un stage effectué à IMT Atlantique, sur le campus de Nantes, au sein du LS2N et de l’équipe MODELIS. Cet environnement scientifique est particulièrement adapté à ce sujet, car il rassemble des compétences en modélisation, optimisation, aide à la décision, logistique, systèmes industriels et intelligence artificielle. Le stage se situe donc au croisement du génie industriel, de la recherche opérationnelle et de l’intelligence artificielle générative.

La problématique principale de ce mémoire peut être formulée de la manière suivante :

Dans quelle mesure les grands modèles de langage peuvent-ils assister la résolution de problèmes d’optimisation combinatoire, et quelles sont les limites de leur fiabilité lorsqu’ils sont évalués à partir de critères objectifs ?

Pour répondre à cette problématique, une démarche expérimentale en deux phases est adoptée. La première phase consiste à évaluer les LLM de manière relativement autonome. Les modèles reçoivent le problème, les données de l’instance et des consignes générales, puis doivent proposer une démarche de résolution et une solution. Cette phase permet d’observer leur comportement spontané, leur capacité de modélisation, leur choix de méthode et leurs erreurs éventuelles.

La deuxième phase introduit un guidage méthodologique. Les modèles ne choisissent plus librement leur méthode de résolution : une heuristique leur est explicitement fournie. Ils doivent alors comprendre cette méthode, l’implémenter, l’appliquer aux données du problème, fournir une solution et vérifier les résultats obtenus. Cette phase permet d’analyser si un guidage heuristique améliore la faisabilité, la qualité objective, la fiabilité et la correction de l’implémentation des solutions générées.

Les expérimentations portent principalement sur trois problèmes classiques d’optimisation combinatoire : le problème du p -médian, le *Set Covering Problem* et l’*Uncapacitated Lot Sizing Problem*. Ces problèmes permettent d’évaluer les LLM sur différentes structures décisionnelles : localisation-affectation, couverture et planification de production multi-périodes. Le *Vehicle Routing Problem with Delivery Options* est également présenté comme un problème logistique plus complexe, permettant de situer le travail dans une perspective plus large liée au routage et à la livraison du dernier kilomètre.

L’évaluation des réponses générées repose sur plusieurs indicateurs : faisabilité de la solution, valeur de la fonction objectif, exactitude du calcul, gap par rapport à une valeur optimale ou de référence, fiabilité de la réponse, correction de l’implémentation et amélioration par rapport à la phase initiale. Ces indicateurs permettent de dépasser une analyse purement qualitative des réponses et d’identifier précisément les forces et les limites des modèles étudiés.

Ce mémoire est organisé comme suit. Le premier chapitre présente l’organisme d’accueil, le cadre institutionnel du stage, le LS2N, l’équipe MODELIS ainsi que le contexte général du sujet. Le deuxième chapitre introduit les fondements de l’optimisation combinatoire, les principales méthodes de résolution, les problèmes étudiés et les liens entre intelligence artificielle et recherche opérationnelle. Les chapitres suivants présentent la méthodologie expérimentale, les résultats obtenus lors des différentes phases d’évaluation, ainsi que l’analyse comparative des performances des LLM.

À travers cette étude, l'objectif est de contribuer à une meilleure compréhension du rôle que peuvent jouer les LLM dans un processus de résolution en optimisation combinatoire. Les résultats attendus ne visent pas seulement à déterminer si les modèles peuvent produire de bonnes solutions, mais surtout à comprendre dans quelles conditions ils peuvent être utiles, quelles erreurs ils commettent, et pourquoi une vérification externe reste indispensable.

Optimisation combinatoire : modélisation, méthodes de résolution et hybridation IA–RO

1.1 Introduction

L'optimisation est un outil central de l'ingénierie et de la recherche opérationnelle : elle formalise un choix à effectuer sous contraintes, en cherchant la meilleure décision au sens d'un objectif quantifiable, tel qu'un coût, une distance, un temps, un risque ou un profit. Dans les systèmes industriels et logistiques, l'optimisation est omniprésente : affecter des ressources, dimensionner des capacités, planifier des opérations, localiser des installations, concevoir des réseaux ou piloter des stocks.

Dans ce mémoire, l'attention se concentre sur l'**optimisation combinatoire**, c'est-à-dire des problèmes où les variables de décision sont discrètes, souvent binaires ou entières, et où l'espace de recherche devient très vaste dès que la taille de l'instance augmente. Dans ce contexte, la résolution exacte peut devenir coûteuse, et des stratégies alternatives, telles que les heuristiques, les métaheuristiques ou les approches hybrides IA–RO, s'imposent souvent [7].

L'objectif de ce chapitre est de fournir un cadre théorique permettant de comprendre :

- les bases de l'optimisation et la typologie des problèmes ;
- la spécificité et la complexité des problèmes combinatoires ;
- les principales méthodes de résolution : exactes, heuristiques et métaheuristiques ;
- les familles industrielles majeures : localisation, couverture, planification de production et routage ;
- la modélisation mathématique des quatre problèmes étudiés ;
- l'hybridation IA–Recherche Opérationnelle et sa pertinence pour l'évaluation des LLM.

Ce chapitre prépare ainsi la compréhension des problèmes soumis aux grands modèles de langage, ainsi que le rôle potentiel de l’hybridation entre intelligence artificielle et Recherche Opérationnelle dans leur résolution.

1.2 Introduction à l’optimisation

1.2.1 Définition générale

Un problème d’optimisation consiste à choisir des variables de décision $\mathbf{x}$ afin de minimiser ou maximiser une fonction objectif $f(\mathbf{x})$, sous contraintes. Une écriture générique est :

$$\min_{\mathbf{x}} f(\mathbf{x}) \quad (1.1)$$

$$\text{s.c. } g_k(\mathbf{x}) \leq 0 \quad \forall k \quad (1.2)$$

$$h_\ell(\mathbf{x}) = 0 \quad \forall \ell \quad (1.3)$$

$$\mathbf{x} \in \mathcal{X} \quad (1.4)$$

Les contraintes traduisent des règles physiques, organisationnelles ou opérationnelles, tandis que $\mathcal{X}$ définit le domaine des variables : continu, entier ou binaire. Cette formulation générale englobe une grande partie des problèmes rencontrés en ingénierie industrielle et en logistique [9].

1.2.2 Minimisation et maximisation

La minimisation est souvent associée à des coûts, tels que la distance, le temps ou les pénalités, tandis que la maximisation vise des profits ou des niveaux de service. Une maximisation peut être transformée en minimisation en changeant le signe de l’objectif :

$$\max f(\mathbf{x}) \equiv \min -f(\mathbf{x})$$

Cette dualité permet d’utiliser des cadres algorithmiques similaires pour les deux types de problèmes.

1.2.3 Rôle des contraintes

Les contraintes structurent fortement le problème. On distingue généralement deux types de contraintes [7] :

- **Contraintes dures** : elles sont inviolables, comme les contraintes de sécurité, de capacité maximale ou de couverture obligatoire. Une violation de ces contraintes rend la solution immédiatement inutilisable.
- **Contraintes souples** : elles peuvent être violées moyennant une pénalité dans la fonction objectif.

En optimisation combinatoire, les contraintes dures sont très fréquentes. C’est pourquoi la notion de **faisabilité** est un critère d’évaluation primordial, tant pour les solveurs classiques que pour les LLM [3].

1.2.4 Solution faisable vs solution optimale

Une **solution faisable** respecte toutes les contraintes. Une **solution optimale** est une solution faisable ayant la meilleure valeur objective possible. Sur des problèmes difficiles, on accepte parfois des solutions proches de l'optimal, c'est-à-dire avec un gap faible, obtenues plus rapidement, à condition de mesurer et de documenter leur qualité [7].

Tableau 1.1 : Distinction entre solution faisable et solution optimale

Type de solution	Contraintes	Objectif
Infaisable	Violées	Non pertinent
Faisable	Respectées	Quelconque
Optimale	Respectées	Meilleur possible
Quasi optimale	Respectées	Gap faible par rapport à la référence

1.3 Typologie des problèmes d'optimisation

1.3.1 Optimisation continue vs discrète

En optimisation continue, les variables sont réelles, c'est-à-dire $x \in \mathbb{R}^n$. En optimisation discrète, les variables appartiennent à un ensemble dénombrable, souvent $\{0, 1\}$ ou $\mathbb{Z}$. Les problèmes combinatoires sont un sous-ensemble majeur de l'optimisation discrète, caractérisé par un espace de solutions fini mais extrêmement vaste [10].

1.3.2 Optimisation linéaire vs non linéaire

Lorsque la fonction objectif et les contraintes sont linéaires, on obtient des formulations de programmation linéaire ou de programmation linéaire en nombres entiers. Ces modèles sont résolus par des algorithmes classiques, tels que le simplexe, les méthodes de points intérieurs ou les méthodes de séparation et évaluation. Les non-linéarités augmentent considérablement la difficulté, notamment en présence de non-convexité ou de minima locaux multiples, sauf dans des cas convexes particuliers.

1.3.3 Déterministe vs stochastique et robuste

- **Déterministe** : tous les paramètres sont connus avec certitude.
- **Stochastique** : certains paramètres sont aléatoires, comme la demande ou les temps de traitement. L'incertitude est généralement modélisée par des distributions de probabilité.
- **Robuste** : on cherche des solutions qui restent bonnes sous des perturbations des paramètres, sans connaître nécessairement leur distribution.

Ces variantes accroissent souvent la difficulté combinatoire en ajoutant des scénarios ou des contraintes supplémentaires. Dans ce mémoire, l'étude se concentre sur des instances déterministes afin de comparer les réponses des LLM dans un cadre contrôlé.

1.3.4 Mono-objectif vs multi-objectifs

Les problèmes multi-objectifs impliquent des compromis entre plusieurs objectifs contradictoires, comme le coût et la qualité de service. La solution est alors un ensemble de compromis, appelé front de Pareto. En pratique, on utilise souvent des pondérations ou des contraintes sur un objectif secondaire pour se ramener à un problème mono-objectif.

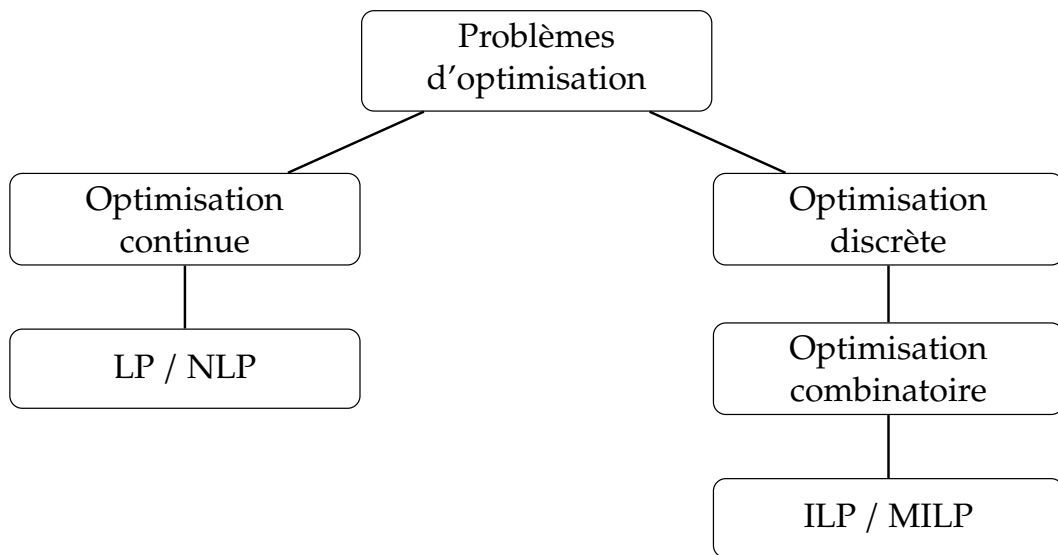


Figure 1.1 : Typologie générale des problèmes d'optimisation

1.4 Optimisation combinatoire : définitions et complexité

1.4.1 Définition formelle

L'optimisation combinatoire traite des problèmes où l'on doit choisir une combinaison de décisions discrètes parmi un ensemble fini de possibilités : sélectionner des sites, couvrir des zones, planifier des lots, ordonner des tâches ou construire des tournées. Formellement, on cherche :

$$\mathbf{x}^* = \arg \min_{\mathbf{x} \in \mathcal{F}} f(\mathbf{x})$$

où $\mathcal{F} \subseteq \{0, 1\}^n$ est l'ensemble des solutions faisables et f est la fonction objectif.

1.4.2 Explosion combinatoire

La difficulté fondamentale de l'optimisation combinatoire est l'**explosion combinatoire** : le nombre de solutions possibles croît très rapidement avec la taille de l'instance. Par exemple :

- choisir p sites parmi n candidats représente $\binom{n}{p}$ possibilités ;
- pour $n = 100$ et $p = 10$, cela représente environ 1.73×10^{13} combinaisons ;
- dans les problèmes de routage, l'ordre de visite, le choix des clients ou des options, ainsi que les contraintes temporelles augmentent encore la taille de l'espace de recherche.

Cette explosion rend l'énumération exhaustive impossible en pratique, d'où la nécessité de méthodes intelligentes de recherche, de bornage et d'amélioration [7].

1.4.3 Théorie de la complexité : NP-difficulté

La théorie de la complexité computationnelle classe les problèmes selon leur difficulté algorithmique [7].

Tableau 1.2 : Classes de complexité et implications pratiques

Classe	Définition	Implication pratique
P	Résolubles en temps polynomial	Efficacement solubles
NP	Vérifiables en temps polynomial	Solution vérifiable rapidement
NP-difficile	Au moins aussi difficile que tout problème NP	Pas d'algorithme polynomial connu
NP-complet	NP-difficile et dans NP	Cas connus : SAT, TSP, etc.

Les problèmes étudiés dans ce mémoire présentent des niveaux de difficulté différents. Le p -médian, le SCP et les variantes de routage telles que le VRPDO sont des problèmes combinatoires difficiles, souvent associés à des formulations en nombres entiers et à des espaces de recherche très vastes. L'ULS, dans sa version standard non capacitaire mono-produit, peut être résolu efficacement par programmation dynamique, mais il reste un problème combinatoire pertinent pour évaluer la capacité des LLM à raisonner sur des décisions de production, de stockage et de lancement de lots.

Cette diversité permet d'évaluer les LLM sur plusieurs structures décisionnelles : localisation, couverture, planification temporelle et routage avec options de livraison.

1.5 Modélisation mathématique : bonnes pratiques de formulation

Dans les problèmes combinatoires, la manière de formuler le modèle est déterminante : deux formulations équivalentes sur le papier peuvent produire des temps de résolution très différents en pratique [3].

1.5.1 Variables, contraintes et interprétation opérationnelle

Une formulation robuste commence par une clarification du rôle de chaque variable :

- **Variables de sélection** : ouvrir ou fermer un site, produire ou ne pas produire, choisir une option de livraison.
- **Variables d'affectation** : affecter un client à un site, un élément à un ensemble ou une demande à une tournée.
- **Variables d'état** : représenter un niveau de stock, un temps d'arrivée ou une charge de véhicule.

Chaque contrainte doit être interprétable opérationnellement : *que garantit-elle ? que rend-elle impossible ?* Cette exigence est utile pour l'expérimentation LLM : des contraintes ambiguës conduisent à des erreurs difficiles à attribuer au modèle ou au prompt [30].

1.5.2 Renforcement de formulation

Certaines contraintes peuvent être redondantes au sens mathématique, mais utiles pour renforcer la relaxation linéaire. Elles réduisent l'espace fractionnaire et accélèrent la recherche. Ce principe explique l'efficacité des approches Branch-and-Cut.

1.5.3 Gestion des symétries

Les symétries correspondent à plusieurs solutions équivalentes ayant la même valeur objectif. Elles augmentent artificiellement la taille de l'arbre de recherche. Des contraintes de brise-symétrie ou des choix de formulation appropriés peuvent réduire ce problème.

1.5.4 Prétraitement et réduction d'instance

Avant la résolution, on peut réduire le problème [7] :

- éliminer des variables dominées, fixées à 0 ou 1 sans perte d'optimalité ;
- réduire des domaines par propagation de contraintes ;

- simplifier des données, par exemple en filtrant des candidats manifestement sous-optimaux.

Dans une étude comparative, ces étapes doivent être documentées car elles modifient la difficulté effective de l'instance.

1.6 Méthodes exactes en optimisation combinatoire

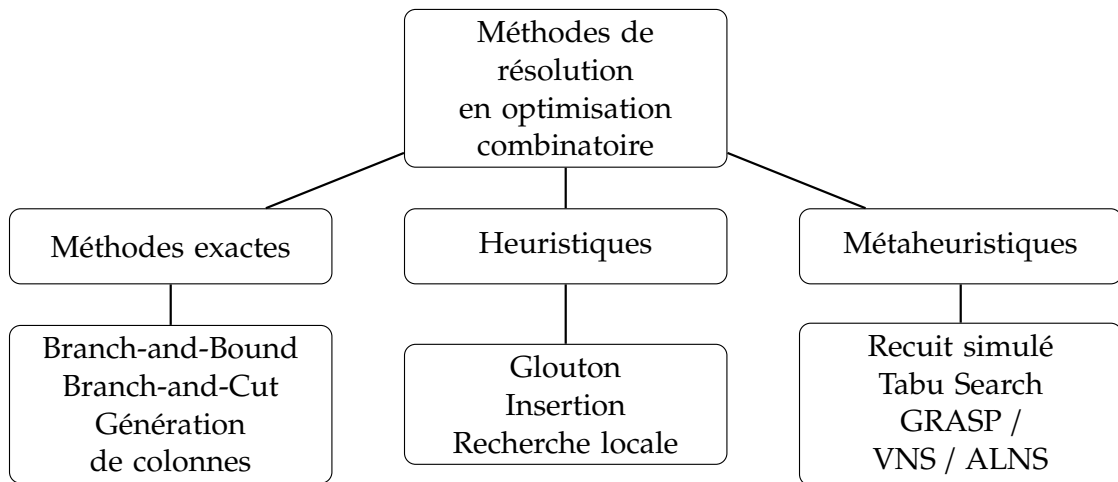


Figure 1.2 : Classification des principales méthodes de résolution en optimisation combinatoire

1.6.1 Programmation linéaire en nombres entiers

La programmation linéaire en nombres entiers, appelée ILP, ou mixte, appelée MILP, permet de modéliser de nombreux problèmes combinatoires avec des fonctions et contraintes linéaires, en imposant certaines variables comme binaires ou entières. C'est une approche centrale en recherche opérationnelle, supportée par des solveurs industriels puissants comme Gurobi, CPLEX et OR-Tools.

Un modèle MILP général s'écrit :

$$\min \quad \mathbf{c}^T \mathbf{x} + \mathbf{d}^T \mathbf{y} \quad (1.5)$$

$$\text{s.c.} \quad A\mathbf{x} + B\mathbf{y} \leq \mathbf{b} \quad (1.6)$$

$$\mathbf{x} \in \mathbb{R}_+^n, \quad \mathbf{y} \in \{0, 1\}^m \quad (1.7)$$

1.6.2 Relaxation linéaire et bornes

La **relaxation linéaire** d'un problème ILP consiste à relâcher les contraintes d'intégrité. Par exemple, une contrainte $y_j \in \{0, 1\}$ devient $0 \leq y_j \leq 1$. La solution optimale de la relaxation fournit une borne inférieure pour une minimisation, utile pour [3] :

- guider l'exploration de l'arbre de recherche ;
- certifier un gap entre la borne et la meilleure solution connue ;

-
- prouver l’optimalité lorsque la borne est égale à la valeur de la meilleure solution.

1.6.3 Branch-and-Bound

Le **Branch-and-Bound** est la méthode exacte fondamentale pour l’ILP. Son principe est le suivant :

1. Résoudre la relaxation linéaire du nœud courant.
2. Si la solution est entière, mettre à jour la meilleure solution connue.
3. Sinon, choisir une variable fractionnaire y_j et créer deux sous-problèmes : $y_j = 0$ et $y_j = 1$.
4. Élaguer les sous-arbres dont la borne est supérieure à la meilleure solution connue.
5. Répéter jusqu’à l’exploration complète de l’arbre.

1.6.4 Branch-and-Cut

Le **Branch-and-Cut** améliore le Branch-and-Bound en ajoutant des **plans coupants**, aussi appelés *cutting planes*. Il s’agit de contraintes valides supplémentaires qui réduisent l’espace fractionnaire sans éliminer de solutions entières. Ces coupes renforcent la relaxation linéaire et accélèrent l’élagage. C’est un pilier des solveurs modernes comme Gurobi et CPLEX.

1.6.5 Génération de colonnes

La **génération de colonnes** est utilisée pour des problèmes avec un très grand nombre de variables. On commence par un sous-ensemble de variables et on génère dynamiquement de nouvelles colonnes si elles peuvent améliorer la solution courante. Elle est particulièrement efficace pour certains problèmes de couverture, de planification et de routage.

1.6.6 Avantages et limites des méthodes exactes

Tableau 1.3 : Avantages et limites des méthodes exactes

Avantages	Limites
Garanties d'optimalité	Coût en temps et en mémoire élevé sur les grandes instances
Reproductibilité	Dépendance forte à la qualité de la formulation
Cadre général MILP	Difficultés sur les structures très complexes
Certification du gap	Explosion combinatoire sur les instances difficiles

1.7 Heuristiques et recherche locale

1.7.1 Heuristiques constructives

Une **heuristique constructive** bâtit une solution à partir de zéro en ajoutant progressivement des éléments selon une règle de priorité. Les approches les plus courantes sont [10] :

- **Glouton** : à chaque étape, choisir l'élément qui améliore le plus l'objectif local.
- **Insertion** : insérer progressivement des éléments dans la solution en minimisant le coût d'insertion.
- **Règles de priorité** : ordonner les éléments selon un critère, par exemple un ratio coût/bénéfice, puis les sélectionner séquentiellement.

Le risque principal est un choix glouton local qui empêche d'atteindre de meilleures solutions globales. C'est pourquoi une phase d'amélioration locale suit souvent la construction.

1.7.2 Recherche locale

La **recherche locale** améliore une solution existante en explorant son voisinage $\mathcal{N}(x)$, c'est-à-dire l'ensemble des solutions accessibles par un petit changement depuis x . L'algorithme accepte un voisin s'il améliore l'objectif, puis s'arrête à un optimum local. Des exemples de voisinages pour les problèmes étudiés sont :

- **p -médian** : swap, c'est-à-dire fermer un site ouvert j , ouvrir un site fermé k , puis réaffecter les clients.
- **SCP** : ajouter ou retirer un ensemble, ou effectuer des échanges de paires.
- **Lot Sizing** : déplacer une production entre périodes, fusionner ou diviser des lots.

1.7.3 Méthodes à population

Les méthodes à population maintiennent un ensemble de solutions et les combinent pour explorer l'espace plus largement. Elles constituent la base des algorithmes génétiques et des algorithmes d'essaims [10].

1.7.4 Réparation de faisabilité

Une stratégie fréquente en optimisation combinatoire est la **réparation** : lorsqu'une solution devient infaisable à la suite d'une perturbation ou d'une mutation, on applique un opérateur de réparation pour restaurer la faisabilité avec un minimum de changements. Cette logique est très proche des boucles agentiques LLM : générer, vérifier, puis réparer [3].

1.8 Métaheuristiques

1.8.1 Définition et objectifs

Les **métaheuristiques** sont des cadres génériques d'optimisation conçus pour explorer efficacement des espaces de solutions très vastes, sans garantie d'optimalité mais avec de bonnes performances pratiques [10]. Elles équilibrent deux mécanismes complémentaires :

- **Intensification** : exploiter les régions prometteuses de l'espace de solutions.
- **Diversification** : explorer de nouvelles régions pour éviter les optima locaux.

1.8.2 Recuit simulé

Le **recuit simulé**, ou *Simulated Annealing*, s'inspire du processus métallurgique de refroidissement lent. Il accepte occasionnellement des solutions dégradantes avec une probabilité :

$$P(\text{accepter}) = e^{-\Delta f/T}$$

où $\Delta f > 0$ est la dégradation et T est la température courante. À mesure que T diminue, le modèle se concentre sur l'intensification. Le recuit simulé est efficace pour échapper aux optima locaux.

1.8.3 Recherche tabou

La **recherche tabou**, ou *Tabu Search*, utilise une mémoire de mouvements récents, appelée liste tabou, pour éviter les cycles et encourager la diversification. Même si un voisin dégrade l'objectif, il peut être accepté s'il n'est pas tabou, permettant ainsi de s'échapper des optima locaux [10].

1.8.4 Algorithmes génétiques

Les **algorithmes génétiques** maintiennent une population de solutions et appliquent des opérateurs inspirés de la génétique :

- **Sélection** : favoriser les meilleures solutions pour la reproduction.
- **Croisement** : combiner deux solutions parentes pour produire des enfants.
- **Mutation** : modifier aléatoirement un élément d'une solution.

1.8.5 GRASP, ILS et VNS

- **GRASP** : *Greedy Randomized Adaptive Search Procedure*. Cette méthode combine constructions aléatoires contrôlées et amélioration locale. Chaque itération produit une solution différente.
- **ILS** : *Iterated Local Search*. Cette méthode alterne perturbations, pour diversifier, et recherche locale, pour intensifier.
- **VNS** : *Variable Neighborhood Search*. Cette méthode change systématiquement de voisinage pour sortir des optima locaux d'un voisinage donné.

1.8.6 ALNS : Adaptive Large Neighborhood Search

L'**ALNS**, ou *Adaptive Large Neighborhood Search*, est une métaheuristique très populaire en routage et en logistique [11]. Elle utilise plusieurs opérateurs de destruction et de réparation, et adapte dynamiquement leur probabilité de sélection selon leurs performances passées. L'ALNS est particulièrement efficace sur le VRP et ses variantes.

Tableau 1.4 : Synthèse des principales métaheuristiques

Méthode	Mécanisme clé	Point fort	Limite
Recuit simulé	Acceptation probabiliste	Échappe aux optima locaux	Sensible au refroidissement
Recherche tabou	Mémoire des mouvements	Évite les cycles	Taille de la liste tabou à régler
Algorithmes génétiques	Population et croisement	Exploration large	Coût computationnel
GRASP	Construction aléatoire et recherche locale	Simple à implémenter	Qualité variable
ILS	Perturbation et recherche locale	Efficace sur structures régulières	Perturbation à calibrer
VNS	Changement de voisinage	Robuste aux optima locaux	Choix des voisinages
ALNS	Opérateurs adaptatifs	Très efficace en routage	Nombreux paramètres

1.8.7 LLM et métaheuristiques

Une revue structurée récente montre que les LLM peuvent jouer un rôle actif dans la conception et l'amélioration des métaheuristiques [10] :

- génération automatique d'opérateurs de voisinage ;
- adaptation dynamique des paramètres, tels que la température ou la taille du voisinage ;
- proposition de nouvelles stratégies d'intensification et de diversification.

Des travaux spécifiques ont exploré la découverte automatique d'opérateurs heuristiques par LLM sur le VRP [11], montrant des résultats compétitifs avec des opérateurs conçus manuellement.

1.9 Les trois problèmes étudiés : modélisations mathématiques

Avant de présenter les modèles étudiés, il est nécessaire de justifier leur choix. Dans ce travail, les trois problèmes retenus ont été sélectionnés afin de couvrir des structures classiques et représentatives de l'optimisation combinatoire. Le choix s'est porté sur le problème du p -median, le problème de couverture d'ensembles (*Set Covering Problem*) et le problème de lot-sizing non capacitaire (*Uncapacitated Lot Sizing Problem*), car ils présentent des caractéristiques différentes en termes de modélisation, de contraintes et de nature décisionnelle.

Le problème du p -median permet d'évaluer la capacité des LLM à traiter un problème de localisation et d'affectation, où il s'agit de sélectionner des emplacements tout en minimisant une distance totale. Le problème de couverture d'ensembles permet, quant à lui, d'analyser leur comportement face à un problème de sélection sous contraintes de couverture, souvent utilisé dans plusieurs applications industrielles et logistiques. Enfin, le problème de lot-sizing introduit une dimension temporelle, avec des décisions de production et de stockage à prendre sur plusieurs périodes.

1.9.1 Problème du p -médian

Description et applications

Le problème du p -médian non capacitaire, ou UPMP, consiste à sélectionner exactement p sites parmi n sites candidats, et à affecter chaque client au site sélectionné le plus proche, afin de minimiser la somme totale des distances ou coûts d'affectation. Ce problème modélise des décisions stratégiques en logistique [7] :

- localisation d'entrepôts et de centres de distribution ;
- implantation d'agences de service ;
- conception de réseaux de transport.

Formulation mathématique

Soient I l'ensemble des clients et J l'ensemble des sites candidats. d_{ij} désigne la distance ou le coût entre le client i et le site j , et p le nombre de sites à ouvrir.

Variables de décision :

- $y_j \in \{0, 1\}$: vaut 1 si le site j est ouvert, 0 sinon.
- $x_{ij} \in \{0, 1\}$: vaut 1 si le client i est affecté au site j , 0 sinon.

Modèle MILP :

$$\min \sum_{i \in I} \sum_{j \in J} d_{ij} x_{ij} \quad (1.8)$$

$$\text{s.c.} \quad \sum_{j \in J} x_{ij} = 1 \quad \forall i \in I \quad (1.9)$$

$$x_{ij} \leq y_j \quad \forall i \in I, \forall j \in J \quad (1.10)$$

$$\sum_{j \in J} y_j = p \quad (1.11)$$

$$x_{ij} \in \{0, 1\}, y_j \in \{0, 1\} \quad \forall i \in I, \forall j \in J \quad (1.12)$$

Interprétation des contraintes :

- (1.9) : chaque client est affecté à exactement un site.
- (1.10) : un client ne peut être affecté qu'à un site ouvert.
- (1.11) : exactement p sites sont ouverts.

Complexité et instances

Le p -médian est un problème combinatoire difficile lorsque p est une donnée du problème. Les instances de référence proviennent notamment de l'OR-Library, avec des tailles variées permettant de tester la robustesse des méthodes de résolution et des réponses générées par les LLM.

1.9.2 Set Covering Problem

Description et applications

Le problème de couverture par ensembles, ou SCP, consiste à sélectionner un sous-ensemble de coût minimal parmi des ensembles candidats, de sorte que chaque élément d'un univers soit couvert par au moins un ensemble sélectionné. Ses applications en logistique et en planification sont nombreuses [31] :

- planification de tournées de véhicules de service ;

- couverture de zones géographiques par des ressources d'urgence ;
- affectation de personnel à des tâches ;
- conception de réseaux de communication.

Formulation mathématique

Soient $U = \{1, \dots, m\}$ l'ensemble des éléments à couvrir et $S = \{1, \dots, n\}$ l'ensemble des ensembles candidats. $a_{us} = 1$ si l'ensemble s couvre l'élément u , et $c_s \geq 0$ est le coût de sélection de l'ensemble s .

Variable de décision :

- $z_s \in \{0, 1\}$: vaut 1 si l'ensemble s est sélectionné, 0 sinon.

Modèle ILP :

$$\min \sum_{s \in S} c_s z_s \quad (1.13)$$

$$\text{s.c.} \quad \sum_{s \in S} a_{us} z_s \geq 1 \quad \forall u \in U \quad (1.14)$$

$$z_s \in \{0, 1\} \quad \forall s \in S \quad (1.15)$$

Interprétation des contraintes :

- (1.14) : chaque élément de U doit être couvert par au moins un ensemble sélectionné. C'est la contrainte de couverture complète, inviolable dans le problème standard.

Complexité et instances

Le SCP est un problème NP-difficile. Les instances OR-Library, notamment les familles de type scp, proposent des matrices de couverture de tailles variées, souvent représentées en format creux afin de réduire la taille des fichiers.

1.9.3 Uncapacitated Lot Sizing

Description et applications

Le problème de dimensionnement de lots non capacitaire, ou ULS, est un problème classique de planification de production sur un horizon multi-périodes T . Il s'agit de déterminer, pour chaque période t , les quantités à produire x_t , le stock à maintenir I_t et la décision de lancement de production y_t , afin de satisfaire une demande d_t connue tout en minimisant les coûts totaux [10]. Ses applications incluent :

- planification de la production manufacturière ;
- gestion des stocks et des approvisionnements ;
- optimisation des coûts de setup dans les lignes de production.

Formulation mathématique

Soient $T = \{1, \dots, H\}$ l'horizon de planification, d_t la demande à la période t , f_t le coût de setup, h_t le coût unitaire de stockage, p_t le coût unitaire de production, et M une grande constante.

Variables de décision :

- $x_t \geq 0$: quantité produite à la période t .
- $I_t \geq 0$: stock en fin de période t .
- $y_t \in \{0, 1\}$: vaut 1 si une production a lieu à la période t .

Modèle MILP :

$$\min \sum_{t \in T} (p_t x_t + f_t y_t + h_t I_t) \quad (1.16)$$

$$\text{s.c. } I_{t-1} + x_t = d_t + I_t \quad \forall t \in T \quad (1.17)$$

$$x_t \leq M \cdot y_t \quad \forall t \in T \quad (1.18)$$

$$x_t \geq 0, I_t \geq 0, y_t \in \{0, 1\} \quad \forall t \in T \quad (1.19)$$

$$I_0 = 0 \quad (1.20)$$

Interprétation des contraintes :

- (1.17) : équation de bilan des stocks. Le stock entrant plus la production est égal à la demande plus le stock sortant.
- (1.18) : contrainte *big-M*. La production n'est possible que si un setup a lieu, c'est-à-dire si $y_t = 1$.
- (1.20) : stock initial nul.

Complexité et instances

Le ULS est un problème NP-difficile. Les instances du depot Github proposent pour chaque période, la demande ainsi que les coûts associés à la production, au lancement et au stockage. Ce problème introduit une dimension temporelle, car les décisions de production et de stockage prises à une période influencent directement les périodes suivantes.

Tableau 1.5 : Comparaison des trois problèmes étudiés

Critère	p -médian	SCP	ULS
Type	Localisation-affectation	Couverture	Planification multi-périodes
Variables	Binaires (x_{ij}, y_j)	Binaires (z_s)	Mixtes (x_t, I_t, y_t)
Difficulté	Problème combinatoire difficile	NP-difficile	Résoluble efficacement dans sa version standard non capacitaire
Source des instances	OR-Library	OR-Library	Benchmarks ULS
Objectif	Minimiser la distance totale d'affectation	Minimiser le coût de couverture	Minimiser les coûts de production, de setup et de stockage
Contrainte clé	Affecter chaque client à un site ouvert	Couvrir chaque élément au moins une fois	Respecter le bilan des stocks

1.10 Hybridation IA–Recherche Opérationnelle

1.10.1 Pourquoi hybrider ?

Les méthodes exactes garantissent l’optimalité mais peuvent avoir un coût computationnel élevé sur les grandes instances. Les heuristiques sont rapides mais nécessitent une expertise métier et ne garantissent pas toujours l’optimalité. L’IA apporte adaptation, apprentissage et automatisation. L’hybridation vise à combiner la rigueur de la Recherche Opérationnelle et la flexibilité de l’IA [31].

Dans le contexte de ce mémoire, cette hybridation est particulièrement importante, car les LLM peuvent proposer des formulations, des solutions ou des raisonnements, mais leurs réponses doivent être vérifiées par des outils fiables. Autrement dit, le LLM peut assister, mais il ne doit pas être considéré comme un solveur autonome.

1.10.2 Formes d’hybridation IA–RO

Tableau 1.6 : Formes d’hybridation IA–RO et exemples [31]

Forme	Description	Exemple
Prédiction + optimisation	Le ML prédit les paramètres et la RO optimise	Prédiction de la demande + lot sizing
Warm start	L’IA génère une solution initiale pour le solveur	LLM + Branch-and-Bound
Apprentissage d’opérateurs	L’IA apprend les meilleurs opérateurs	LLM + ALNS [11]
Pilotage de solveur	L’IA guide certaines décisions du solveur	RL + CPLEX
Aide à la modélisation	Le LLM génère la formulation MILP	LLM + Gurobi [30]
Ancrage par solveur	Le solveur valide et corrige le LLM	SIRL [3]

1.10.3 Place des LLM dans l'hybridation

Les LLM peuvent contribuer à plusieurs niveaux de l'hybridation IA-RO [31, 10] :

- **Traduction** : convertir un énoncé en langage naturel en modèle mathématique formel.
- **Génération de code** : produire un script Python/Gurobi pour résoudre le problème.
- **Proposition d'heuristiques** : suggérer des stratégies de construction ou d'amélioration.
- **Orchestration** : coordonner des outils, tels qu'un solveur ou un vérificateur, dans une boucle agentique.

Dans ce cadre, les LLM ne sont pas considérés comme des solveurs autonomes, mais comme des assistants guidés. Cette distinction est essentielle : un LLM peut produire une réponse convaincante, mais une réponse convaincante n'est pas nécessairement une solution correcte.

1.10.4 LLM + solveur : intégration et ancrage

Dans une approche hybride, le LLM n'est pas utilisé comme un solveur autonome, mais comme un assistant guidé capable de proposer une formulation, une solution candidate ou une stratégie de résolution. Le solveur joue alors le rôle d'outil d'ancrage : il vérifie la faisabilité, calcule la valeur objectif et renvoie un retour objectif au LLM. Cette boucle permet de limiter les erreurs de raisonnement, les affirmations non vérifiées et les fausses déclarations d'optimalité.

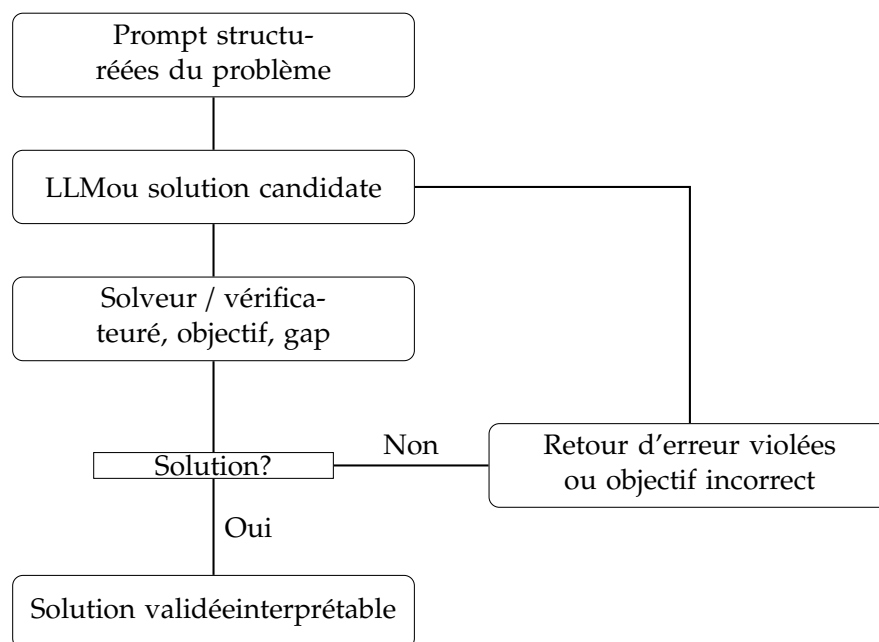


Figure 1.3 : Boucle d'intégration entre LLM et solveur pour l'ancrage des solutions

Ce schéma correspond à une logique d’ancrage par solveur, ou *solver grounding* [3]. Le LLM génère une proposition, mais la validation repose sur un outil externe capable de contrôler rigoureusement les contraintes et la fonction objectif. L’intérêt principal est donc de transformer le LLM en assistant de modélisation et de génération de solutions, tout en évitant de lui accorder une confiance directe lorsqu’il affirme qu’une solution est faisable ou optimale.

1.11 Conclusion

Ce chapitre a présenté les bases de l’optimisation combinatoire nécessaires à la compréhension de nos travaux expérimentaux. Nous avons défini formellement les concepts d’optimisation, détaillé la typologie des problèmes, expliqué la complexité des problèmes combinatoires et passé en revue les principales méthodes de résolution : méthodes exactes, heuristiques et métaheuristiques.

Les formulations mathématiques des problèmes étudiés, à savoir le p -médian, le SCP et l’ULS, ont été présentées avec leurs interprétations opérationnelles. Les problèmes constituent une base expérimentale classique permettant d’évaluer les capacités des LLM sur des structures combinatoires connues : localisation, couverture et planification de production.

La section sur l’hybridation IA–RO a montré que les LLM sont surtout prometteurs lorsqu’ils sont intégrés dans des systèmes contrôlés et outillés [3], automatisant des workflows de modélisation et de résolution [30], ou contribuant à la découverte d’opérateurs heuristiques [11]. La GenAI en Recherche Opérationnelle est un domaine en plein essor [31], et les LLM en métaheuristique font l’objet d’une revue structurée récente [10].

Le chapitre suivant présentera le framework de prompt engineering développé pour soumettre ces problèmes aux LLM évalués, ainsi que la manière dont les réponses générées sont analysées, vérifiées et comparées.

Présentation de l'organisme d'accueil et du contexte du stage

2.1 Introduction

Ce chapitre présente le cadre institutionnel, scientifique et organisationnel dans lequel s'inscrit le stage réalisé au sein d'IMT Atlantique, sur le campus de Nantes. Ce stage constitue une étape importante du projet de fin d'études, car il permet de relier les connaissances acquises en génie industriel, en optimisation et en aide à la décision à un environnement de recherche appliquée.

L'objectif de ce chapitre est de présenter l'organisme d'accueil, le laboratoire dans lequel s'inscrit le stage, ainsi que l'équipe de recherche concernée. Il permet également de situer le sujet du mémoire dans son contexte scientifique, sans reprendre les éléments méthodologiques et techniques qui seront détaillés dans les chapitres suivants.

Le stage s'inscrit dans un contexte marqué par le développement rapide de l'intelligence artificielle générative et des grands modèles de langage, appelés *Large Language Models* (LLM). Ces modèles sont aujourd'hui utilisés dans de nombreux domaines comme outils d'assistance à la rédaction, à la programmation, à l'analyse et à la résolution de problèmes. Dans le domaine de la recherche opérationnelle, leur utilisation soulève une question importante : peuvent-ils assister de manière fiable la modélisation et la résolution de problèmes d'optimisation combinatoire ?

Ce questionnement est particulièrement cohérent avec les thématiques de l'équipe MODELIS, qui portent sur la modélisation, l'optimisation et l'aide à la décision pour la logistique, l'industrie et les services. Le stage s'intègre donc dans un environnement scientifique adapté à l'étude du rôle potentiel des LLM dans des problèmes relevant du génie industriel.

Ce chapitre est organisé comme suit. La première partie présente brièvement IMT Atlantique et le campus de Nantes. La deuxième partie décrit le LS2N, laboratoire d'accueil du stage. La troisième partie présente l'équipe MODELIS et ses thématiques. Une section est ensuite consacrée à l'organisation de la recherche en France, avec une mise en perspective par rapport au contexte algérien. Les parties suivantes situent le sujet du stage, ses apports, les difficultés rencontrées, ainsi que les activités

scientifiques auxquelles j'ai participé durant cette expérience, notamment le séminaire présenté au sein de l'équipe, les échanges avec les doctorants et la découverte de la communauté scientifique liée à la ROADEF.

2.2 Présentation générale d'IMT Atlantique

2.2.1 Identité et positionnement

IMT Atlantique est une grande école d'ingénieurs généralistes française relevant de l'Institut Mines-Télécom. Elle est placée sous la tutelle du ministère chargé de l'industrie et du numérique. L'école se positionne à l'intersection de plusieurs domaines stratégiques : le numérique, l'énergie, l'environnement, les systèmes industriels et l'innovation technologique [17].

Créée le 1er janvier 2017, IMT Atlantique est issue de la fusion de Mines Nantes et Télécom Bretagne [13]. Elle dispose de plusieurs campus, notamment à Brest, Nantes et Rennes. Cette organisation multi-sites lui permet de développer des activités de formation, de recherche et d'innovation en lien avec différents écosystèmes territoriaux.

Dans le cadre de ce mémoire, le campus de Nantes occupe une place particulière, car il est fortement lié aux domaines de l'automatique, de la productique, de l'informatique, de la logistique, des sciences du numérique et de la recherche opérationnelle [16]. Ces domaines sont directement en lien avec le sujet du stage.

2.2.2 Missions principales

Les missions d'IMT Atlantique s'articulent principalement autour de trois axes : la formation, la recherche et l'innovation.

La mission de formation vise à préparer des ingénieurs capables de comprendre et de piloter des systèmes complexes. Ces systèmes combinent souvent des dimensions techniques, économiques, organisationnelles, numériques et environnementales. Cette vision correspond aux besoins actuels des entreprises industrielles et logistiques.

La mission de recherche repose sur l'activité des enseignants-chercheurs de l'école, qui participent à différents laboratoires et unités mixtes de recherche. Les domaines couverts incluent notamment les sciences du numérique, l'automatique, les systèmes énergétiques, l'environnement, la cybersécurité, la robotique, la logistique et la recherche opérationnelle.

Enfin, la mission d'innovation vise à renforcer les liens entre recherche académique, entreprises et société. Cette dimension est importante dans un sujet comme celui de ce mémoire, car l'utilisation des LLM pour l'aide à la décision interroge à la fois les méthodes scientifiques et les usages professionnels futurs.

2.2.3 Lien avec le sujet du stage

Le sujet du stage se situe au croisement de plusieurs domaines présents à IMT Atlantique : le génie industriel, l’optimisation combinatoire, l’intelligence artificielle, la science des données et l’aide à la décision. Il s’agit d’étudier si les LLM peuvent être utilisés comme assistants pour traiter des problèmes nécessitant habituellement des compétences avancées en modélisation mathématique et en recherche opérationnelle.

Cette problématique s’inscrit dans une dynamique actuelle où les outils d’intelligence artificielle sont de plus en plus mobilisés dans les environnements académiques et industriels. Cependant, leur usage dans des tâches de décision reste délicat, car une réponse textuellement cohérente ne garantit pas nécessairement une solution faisable, correcte ou optimale.

2.3 Le campus de Nantes

2.3.1 Localisation et environnement scientifique

Le stage est réalisé sur le campus de Nantes d’IMT Atlantique. Ce campus est situé au nord de la ville, au bord de l’Erdre, dans un environnement universitaire et scientifique dynamique [16]. Sa localisation favorise les interactions avec plusieurs établissements d’enseignement supérieur et de recherche présents dans la métropole nantaise.

La ville de Nantes constitue un environnement favorable à la recherche appliquée et à l’innovation. Elle accueille un tissu universitaire important, des laboratoires reconnus, des entreprises technologiques et des structures d’accompagnement de projets innovants. Cette proximité entre acteurs académiques et économiques facilite les collaborations et le transfert de connaissances.

2.3.2 Un cadre adapté au stage

Le campus de Nantes offre un cadre cohérent pour un stage portant sur l’optimisation combinatoire et l’intelligence artificielle. Il réunit des compétences en modélisation, informatique, automatique, productique, logistique et aide à la décision.

Dans le cadre du stage, cet environnement permet de travailler au contact de chercheurs spécialisés dans la recherche opérationnelle, les systèmes de production, la logistique et les méthodes d’optimisation. Il permet également de mieux comprendre la manière dont une problématique scientifique est construite : formulation d’une question de recherche, étude bibliographique, choix méthodologiques, expérimentation, analyse critique et rédaction.

Le stage ne se limite donc pas à une application technique. Il constitue aussi une immersion dans un environnement de recherche, avec ses exigences de rigueur, de traçabilité, de justification des choix et de reproductibilité.

2.4 Présentation du LS2N

2.4.1 Identité du laboratoire

Le Laboratoire des Sciences du Numérique de Nantes, plus connu sous l'acronyme LS2N, est une unité mixte de recherche référencée UMR6004. Il regroupe des activités de recherche dans le domaine des sciences du numérique, au sens large : informatique, automatique, robotique, traitement du signal, traitement de l'image, sciences des données, décision, ergonomie, interaction humain-machine et systèmes distribués [2].

Le LS2N est associé à plusieurs institutions, notamment Centrale Nantes, Nantes Université, IMT Atlantique, le CNRS et Inria [2]. Cette organisation multi-tutelles reflète le caractère interdisciplinaire du laboratoire et favorise les collaborations entre différentes communautés scientifiques.

2.4.2 Organisation générale

Le LS2N rassemble des chercheurs, enseignants-chercheurs, doctorants, ingénieurs, post-doctorants et personnels administratifs. Il est organisé autour de plusieurs pôles scientifiques et équipes de recherche couvrant un large spectre des sciences du numérique.

10

Tableau 2.1 : Synthèse de l'organisation du LS2N

Élément	Description
Nom complet	Laboratoire des Sciences du Numérique de Nantes
Référence	UMR6004
Institutions associées	Centrale Nantes, Nantes Université, IMT Atlantique, CNRS, Inria
Domaines principaux	Informatique, automatique, robotique, données, décision, interaction humain-machine
Lien avec le stage	Recherche opérationnelle, optimisation, aide à la décision et intelligence artificielle

Cette structuration permet au laboratoire de traiter des problématiques scientifiques variées, allant des méthodes fondamentales aux applications industrielles. Le sujet du stage s'inscrit dans cette logique, car il combine des outils récents d'intelligence artificielle avec des problématiques classiques de modélisation et de décision.

2.4.3 Lien avec les sciences de la décision

Le LS2N développe des recherches liées à l'aide à la décision, à l'optimisation, aux systèmes complexes et aux sciences des données. Ces thématiques sont particulièrement importantes pour les systèmes industriels et logistiques, qui nécessitent souvent de prendre des décisions sous contraintes multiples.

Dans ce contexte, le stage trouve sa place dans une réflexion plus large sur l'utilisation de nouveaux outils numériques pour assister la décision. Les LLM peuvent faciliter la compréhension d'un problème, la formulation d'un modèle ou la génération d'un code de résolution. Toutefois, leur utilisation nécessite une vérification rigoureuse, surtout lorsque les résultats obtenus doivent servir de support à une décision.

2.5 Présentation de l'équipe MODELIS

2.5.1 Identité et rattachement

Le stage est effectué au sein de l'équipe MODELIS. L'acronyme MODELIS signifie *Modélisation, Optimisation et DEcision pour la Logistique, l'Industrie et les Services*. Cette équipe appartient au LS2N et s'inscrit dans les activités du département Automatique, Productique et Informatique d'IMT Atlantique, appelé DAPI [15].

MODELIS développe des méthodes de recherche opérationnelle pour l'optimisation et l'aide à la décision dans les domaines de la production, de la logistique, du transport, de l'industrie et des services [14]. Ses travaux portent à la fois sur des problèmes théoriques difficiles et sur des problèmes appliqués issus de situations industrielles réelles.

2.5.2 Axes scientifiques

Les activités de MODELIS peuvent être résumées autour de trois axes principaux : la modélisation, l'optimisation et l'aide à la décision.

La **modélisation** consiste à traduire une situation réelle ou un problème industriel en modèle mathématique. Cette étape est essentielle, car la qualité de la formulation influence directement la pertinence de la solution obtenue.

L'**optimisation** vise ensuite à rechercher une solution performante, voire optimale, en tenant compte des contraintes du problème. Les méthodes mobilisées peuvent être exactes, heuristiques, métaheuristiques ou hybrides, selon la nature du problème et la taille des instances.

Enfin, l'**aide à la décision** cherche à rendre les résultats exploitables par les décideurs. Une solution ne doit pas seulement être calculée : elle doit aussi être compréhensible, justifiable et adaptée au contexte opérationnel.

2.6 Organisation de la recherche en France et comparaison avec le contexte algérien

2.6.1 Structure générale de la recherche en France

Le stage réalisé à IMT Atlantique m'a permis de comprendre l'organisation de la recherche scientifique en France. Cette organisation repose sur une articulation

entre les établissements d'enseignement supérieur, les organismes nationaux de recherche, les laboratoires, les équipes scientifiques et les écoles doctorales.

En France, un laboratoire peut être rattaché à plusieurs tutelles. C'est notamment le cas des unités mixtes de recherche, qui associent plusieurs établissements ou organismes autour d'une même structure scientifique. Le LS2N constitue un exemple de cette organisation, puisqu'il est associé à plusieurs institutions telles que Centrale Nantes, Nantes Université, IMT Atlantique, le CNRS et Inria. Cette organisation multi-tutelles favorise les collaborations entre chercheurs issus de différentes institutions et permet de mutualiser les compétences, les moyens humains et les ressources scientifiques.

La formation doctorale occupe également une place importante dans cette structuration. Le doctorant est inscrit dans un établissement habilité à délivrer le doctorat, tout en étant rattaché à une école doctorale et à une unité de recherche. L'école doctorale assure un cadre de formation, de suivi et d'accompagnement, tandis que le laboratoire constitue le lieu principal de réalisation des travaux de recherche.

Cette organisation m'a permis de mieux comprendre le fonctionnement d'un environnement de recherche structuré. Le travail scientifique ne se limite pas à la réalisation d'expériences ou à la rédaction d'un mémoire. Il s'inscrit dans un écosystème composé d'équipes, de séminaires, de collaborations, de publications, d'échanges scientifiques et de projets de recherche.

2.6.2 Comparaison avec le contexte algérien

Cette immersion m'a également permis de comparer, de manière générale, la structuration de la recherche en France avec celle que j'ai connue en Algérie. En Algérie, la recherche scientifique est principalement organisée autour des universités, des écoles supérieures, des centres de recherche, des unités de recherche et des laboratoires de recherche. Ces structures jouent un rôle essentiel dans la formation, l'encadrement des étudiants et le développement des travaux scientifiques.

La comparaison entre les deux contextes montre que les deux systèmes accordent une place importante à l'université et aux laboratoires. Cependant, l'organisation française se distingue par une forte logique de structuration en unités mixtes, souvent rattachées à plusieurs institutions, ainsi que par une intégration très visible des doctorants dans les équipes de recherche, les séminaires et les réseaux scientifiques. Cette immersion m'a ainsi permis de mieux comprendre la place du laboratoire comme espace collectif de recherche, mais aussi le rôle des échanges scientifiques dans la construction progressive d'un travail académique.

Dans le cadre de mon parcours, cette comparaison a été particulièrement enrichissante. Ma formation en génie industriel à l'École Supérieure en Sciences Appliquées de Tlemcen m'a apporté les bases nécessaires en optimisation, logistique, gestion de production et aide à la décision. Le stage à IMT Atlantique m'a permis de prolonger ces acquis dans un environnement de recherche appliquée, en découvrant une organisation scientifique différente et complémentaire.

2.7 Contexte général du stage

2.7.1 Sujet du stage

Le stage porte sur l'évaluation des grands modèles de langage comme outils d'aide à la résolution de problèmes d'optimisation combinatoire. Plus précisément, il s'agit d'analyser dans quelle mesure des LLM peuvent produire des réponses exploitables pour des problèmes classiques de logistique, de production et de routage.

Le sujet peut être formulé de la manière suivante :

Évaluer la capacité des grands modèles de langage à assister la résolution de problèmes d'optimisation combinatoire appliqués à la logistique et à la production.

Cette problématique est actuelle, car les LLM sont de plus en plus utilisés dans des contextes académiques et professionnels. Ils peuvent expliquer des concepts, produire du code, structurer une démarche ou proposer des solutions. Toutefois, lorsqu'ils sont utilisés pour des problèmes d'optimisation, leur fiabilité doit être contrôlée de manière rigoureuse.

2.7.2 Motivation scientifique

La motivation scientifique du stage repose sur l'écart entre la capacité apparente des LLM à produire des raisonnements convaincants et la nécessité, en optimisation, de fournir des solutions vérifiables. Une solution d'optimisation ne peut pas être évaluée uniquement par la qualité de son explication. Elle doit respecter des contraintes, avoir une valeur objectif correctement calculée et, lorsque cela est possible, être comparée à une référence.

Cette exigence rend l'étude des LLM particulièrement intéressante dans le domaine de l'optimisation combinatoire. Elle permet d'aller au-delà d'une simple appréciation qualitative des réponses et d'évaluer objectivement leur validité.

2.7.3 Motivation industrielle

Du point de vue industriel, l'intérêt du sujet est également important. De nombreuses entreprises rencontrent des problèmes d'optimisation dans leurs activités quotidiennes : localisation, transport, production, planification, gestion des stocks ou affectation de ressources. Pourtant, toutes ne disposent pas d'experts en recherche opérationnelle ou de compétences avancées en programmation.

Les LLM pourraient donc représenter une interface accessible entre des utilisateurs non experts et des méthodes d'optimisation plus complexes. Ils pourraient aider à formuler un problème, expliquer une méthode ou générer un premier code de résolution.

Cependant, cette possibilité doit être abordée avec prudence. Un LLM peut produire une réponse plausible mais incorrecte. Le stage vise donc à analyser non seulement ce que les LLM sont capables de produire, mais aussi les limites de leurs réponses lorsqu'elles sont confrontées à une vérification rigoureuse.

2.7.4 Positionnement dans le mémoire

Ce chapitre présente le cadre du stage. Les aspects techniques, comme les problèmes étudiés, le protocole expérimental, les prompts, les indicateurs d'évaluation et l'analyse des résultats, sont détaillés dans les chapitres suivants afin d'éviter les répétitions.

Le rôle de ce chapitre est donc de situer le travail dans son environnement institutionnel et scientifique : IMT Atlantique, le LS2N, l'équipe MODELIS et les enjeux généraux du sujet.

2.8 Apports du stage

2.8.1 Apports scientifiques

Le stage apporte une contribution à l'étude des LLM dans un contexte où les réponses produites doivent être vérifiables. Contrairement à des tâches purement rédactionnelles, les problèmes d'optimisation imposent une validation objective : une solution respecte ou non les contraintes, et sa valeur peut être recalculée.

Cette caractéristique rend le sujet intéressant d'un point de vue scientifique, car elle permet d'évaluer les LLM sur leur capacité à produire des décisions structurées, et non seulement des explications textuelles.

2.8.2 Apports méthodologiques

Le stage permet également de construire une démarche d'évaluation rigoureuse des réponses générées par les LLM. Cette démarche repose sur la comparaison entre les réponses produites, les contraintes des problèmes et les valeurs de référence obtenues par des méthodes classiques.

L'apport méthodologique réside donc dans la mise en place d'un cadre permettant de distinguer une réponse simplement plausible d'une réponse réellement exploitable.

2.8.3 Apports pour le génie industriel

Pour le génie industriel, le stage propose une réflexion sur l'utilisation des outils d'intelligence artificielle générative dans l'aide à la décision. Les ingénieurs industriels sont régulièrement confrontés à des problèmes complexes nécessitant des compromis entre coûts, délais, ressources et contraintes opérationnelles.

Les LLM pourraient devenir des assistants utiles pour expliquer des modèles, générer des formulations ou faciliter l'accès à des méthodes d'optimisation. Toutefois, leur utilisation doit rester encadrée par des mécanismes de vérification.

2.8.4 Apports personnels et professionnels

Sur le plan personnel et professionnel, le stage permet de développer plusieurs compétences : compréhension des problèmes d'optimisation combinatoire, analyse critique des résultats, utilisation d'outils de vérification, organisation d'une démarche expérimentale et rédaction scientifique.

Il constitue également une immersion dans un environnement de recherche, ce qui permet de mieux comprendre les exigences associées à un travail académique : rigueur, autonomie, traçabilité, justification des choix et recul critique.

2.8.5 Immersion dans un environnement scientifique

Au-delà du travail technique réalisé dans le cadre du mémoire, le stage a constitué une véritable immersion dans un environnement scientifique. Cette immersion s'est traduite par la participation aux échanges de l'équipe, la présentation de mon travail lors d'un séminaire, ainsi que des discussions avec des doctorants travaillant sur des thématiques proches de l'optimisation, de l'aide à la décision et des systèmes industriels.

La présentation de mon séminaire au sein de l'équipe a été une étape importante de ce stage. Elle m'a permis de présenter le contexte de mon travail, la problématique étudiée, la méthodologie suivie et les premiers résultats obtenus. Cet exercice a été particulièrement formateur, car il m'a obligée à structurer mon discours, à clarifier mes choix méthodologiques et à défendre la pertinence de mon approche devant un public scientifique. Les remarques et questions reçues m'ont permis d'améliorer ma réflexion et de mieux identifier les points à approfondir dans la suite du mémoire.

Les échanges avec les doctorants ont également constitué un apport important. Ils m'ont permis de mieux comprendre la réalité du travail doctoral, notamment la construction progressive d'une problématique, l'importance de la revue de littérature, la nécessité de justifier les choix méthodologiques et le rôle central de la validation expérimentale. Ces discussions ont aussi été utiles pour mieux situer mon propre travail dans un cadre de recherche plus large.

Le stage m'a également permis de découvrir la communauté scientifique liée à la recherche opérationnelle, notamment à travers la ROADEF. Cette communauté joue un rôle important dans la diffusion des travaux en optimisation, aide à la décision et génie industriel. La découverte de cette communauté et de ses événements scientifiques m'a permis de mieux comprendre l'importance des conférences dans la valorisation des travaux de recherche, la discussion des résultats et la construction de collaborations scientifiques.

Ainsi, cette expérience ne s'est pas limitée à la réalisation d'un projet technique. Elle m'a permis de découvrir concrètement le fonctionnement d'un environnement de recherche, depuis le travail individuel d'analyse jusqu'aux échanges collectifs avec les chercheurs, doctorants et membres de l'équipe.

2.9 Difficultés rencontrées

2.9.1 Complexité du sujet

La première difficulté concerne la nature interdisciplinaire du sujet. Le stage se situe à la frontière entre intelligence artificielle, recherche opérationnelle, génie industriel et analyse expérimentale. Il nécessite donc de mobiliser plusieurs types de compétences : compréhension théorique, modélisation, programmation, vérification et interprétation des résultats.

2.9.2 Fiabilité des réponses des LLM

Une difficulté importante concerne la fiabilité des réponses produites par les LLM. Ces modèles peuvent fournir des explications convaincantes tout en commettant des erreurs dans les calculs, les contraintes ou l'interprétation du statut d'une solution. Cette difficulté renforce la nécessité de ne pas se limiter à l'apparence de la réponse. Elle impose de vérifier systématiquement les solutions proposées.

2.9.3 Traçabilité expérimentale

Le travail expérimental exige également une bonne organisation. Il est nécessaire de conserver les prompts utilisés, les réponses obtenues, les modèles interrogés, les dates d'accès et les résultats de vérification. Cette traçabilité est indispensable pour garantir la cohérence et la reproductibilité de l'étude.

2.10 Lien entre le stage et la formation

Le stage est directement lié à la formation en génie industriel, notamment à travers les notions d'optimisation, de planification, de logistique, de gestion de production et d'aide à la décision. Ces notions sont essentielles pour concevoir, améliorer et piloter des systèmes industriels performants.

La formation en génie industriel fournit les bases nécessaires pour comprendre les problèmes étudiés : formulation d'objectifs, identification des contraintes, modélisation des flux, gestion des ressources et analyse de la performance. Le stage permet d'approfondir ces connaissances dans un contexte de recherche.

Il permet également d'intégrer une dimension nouvelle : l'intelligence artificielle générative. Cette dimension est de plus en plus importante dans les métiers de l'ingénierie, car les outils d'IA transforment progressivement les pratiques professionnelles. Savoir les utiliser, mais aussi savoir en évaluer les limites, devient une compétence importante pour un futur ingénieur.

Ainsi, le stage constitue un lien entre la formation académique, la recherche scientifique et les besoins industriels actuels.

2.11 Conclusion

Ce chapitre a présenté le cadre général du stage réalisé au sein d'IMT Atlantique, sur le campus de Nantes. L'école constitue un environnement académique et scientifique adapté à un travail portant sur l'optimisation, l'intelligence artificielle et l'aide à la décision.

Le LS2N, laboratoire d'accueil, offre un cadre interdisciplinaire centré sur les sciences du numérique. L'équipe MODELIS, dans laquelle s'inscrit le stage, développe des méthodes de modélisation, d'optimisation et d'aide à la décision pour la logistique, l'industrie et les services.

Le stage porte sur l'évaluation des LLM comme assistants pour la résolution de problèmes d'optimisation combinatoire. Cette problématique est à la fois scientifique et industrielle, car elle interroge la capacité des outils d'intelligence artificielle générative à produire des réponses utiles dans des contextes où la faisabilité, la fiabilité et la vérification sont essentielles.

Ce chapitre constitue ainsi une présentation du cadre pratique, institutionnel et scientifique du mémoire. Il met également en évidence l'apport de l'immersion dans un environnement de recherche, à travers les échanges avec les doctorants, la présentation d'un séminaire et la découverte de la communauté scientifique liée à la recherche opérationnelle. Les chapitres suivants détailleront les concepts théoriques, la méthodologie expérimentale, les résultats obtenus et l'analyse critique des performances des LLM.

Evaluation guidée des LLM sur des problèmes d'optimisation

3.1 Introduction

Après une première phase expérimentale dans laquelle les grands modèles de langage ont été évalués de manière relativement autonome, cette deuxième phase vise à analyser leur comportement dans un cadre plus guidé. L'objectif n'est plus seulement de demander aux LLM de résoudre directement un problème d'optimisation combinatoire, mais de leur fournir une méthode heuristique précise, de leur expliquer son fonctionnement, puis de leur demander de l'implémenter afin de résoudre les instances étudiées.

Cette phase s'inscrit donc dans une logique d'assistance guidée. Contrairement à la phase 1, où les LLM devaient choisir eux-mêmes leur stratégie de résolution, la phase 2 cherche à déterminer si leurs performances peuvent s'améliorer lorsqu'ils sont orientés vers une méthode déjà identifiée comme pertinente. L'idée principale est d'évaluer leur capacité non seulement à comprendre une heuristique, mais aussi à la traduire correctement en code, à l'appliquer aux données du problème et à produire des résultats vérifiables.

Le problème du p -médian constitue la première application de cette phase guidée. Ce choix s'explique par le fait qu'il a déjà été étudié dans la première phase et qu'il permet donc une comparaison directe entre les résultats autonomes et les résultats guidés. Les sections suivantes présentent la méthodologie générale de la phase 2, avant d'aborder l'expérimentation appliquée au problème du p -médian.

3.2 Principe général de la phase 2

La phase 2 repose sur une logique différente de celle de la phase 1. Dans la première phase, les LLM recevaient le problème, les données de l'instance et les consignes générales, puis devaient proposer eux-mêmes une méthode de résolution. Cette approche permettait d'évaluer leur comportement autonome face à un problème d'optimisation combinatoire.

Cependant, les résultats de la phase 1 ont montré que certains modèles pouvaient produire des solutions infaisables, des valeurs objectives incorrectes ou des solutions faisables mais éloignées de l'optimum. Pour cette raison, la phase 2 introduit un guidage méthodologique plus fort : les LLM reçoivent une heuristique précise, inspirée des méthodes utilisées par les modèles ayant obtenu les meilleurs résultats dans la phase 1.

L'objectif de cette phase est donc de répondre à la question suivante :

Le guidage méthodologique améliore-t-il la capacité des LLM à produire des solutions faisables, fiables et de bonne qualité pour des problèmes d'optimisation combinatoire ?

Cette question permet d'analyser le passage d'un usage autonome des LLM vers un usage plus contrôlé, dans lequel le modèle agit comme un assistant chargé de comprendre, implémenter et appliquer une méthode donnée.

3.3 Sélection des modèles évalués

Tous les LLM testés en phase 1 ne sont pas nécessairement repris dans cette deuxième phase. La phase 2 cible principalement les modèles ayant rencontré des difficultés lors de la première phase. Plus précisément, les modèles retenus sont ceux qui ont produit :

- des solutions infaisables ;
- ou des solutions faisables mais avec un gap supérieur à 5% par rapport à la solution optimale ou de référence.

Ce choix permet de concentrer l'analyse sur les modèles qui disposent d'une marge d'amélioration significative. Il serait peu pertinent de guider à nouveau un modèle ayant déjà obtenu une solution optimale ou quasi optimale en phase 1, sauf si l'objectif était uniquement de confirmer sa stabilité. Ici, l'intérêt est plutôt de vérifier si le guidage peut aider les modèles les moins performants à corriger leurs erreurs, à améliorer la faisabilité de leurs solutions ou à réduire leur écart à l'optimum.

Les modèles ayant fourni des résultats optimaux ou avec un gap inférieur à 5% en phase 1 servent indirectement de référence méthodologique. Les heuristiques qu'ils ont mobilisées ou suggérées sont analysées afin de construire les consignes de la phase 2. Autrement dit, la phase 2 ne guide pas les modèles de manière arbitraire : elle s'appuie sur les stratégies qui ont déjà montré un intérêt dans la première phase.

3.4 Construction du guidage heuristique

Le guidage proposé aux LLM repose sur les heuristiques observées lors de la première phase. Les méthodes utilisées ou proposées par les modèles les plus performants ont été examinées afin d'identifier des stratégies de résolution pertinentes pour chaque problème étudié.

À partir de cette analyse, une heuristique est sélectionnée puis explicitement fournie aux modèles évalués. Le prompt ne se limite donc pas à demander une solution finale. Il décrit la méthode à suivre, ses principales étapes et les contraintes à respecter. Le LLM doit ensuite :

1. comprendre l'heuristique proposée ;
2. l'implémenter sous forme de code ;
3. appliquer ce code à l'instance étudiée ;
4. retourner la solution obtenue et sa valeur objective ;
5. vérifier la faisabilité de la solution ;
6. expliquer brièvement la démarche suivie.

Cette approche permet d'évaluer une compétence différente de celle observée en phase 1. Le modèle n'est plus jugé uniquement sur sa capacité à choisir spontanément une méthode, mais aussi sur sa capacité à exécuter correctement une méthode imposée. C'est précisément ce qui permet d'analyser l'apport du guidage.

3.5 Configuration des prompts

Dans cette deuxième phase, les prompts utilisés sont des prompts guidés. Contrairement à la phase 1, où les LLM disposaient d'une plus grande liberté dans le choix de la méthode de résolution, les prompts de phase 2 imposent une heuristique précise et demandent au modèle de l'appliquer explicitement.

Tous les prompts de cette phase sont rédigés avec une **spécification de rôle**. Chaque prompt commence par l'instruction suivante :

You are an expert in combinatorial optimization and heuristic methods.

Ce choix permet de conserver la configuration ayant donné les meilleurs résultats dans la première phase, tout en orientant les modèles vers un comportement plus technique. Les prompts sont formulés en anglais afin de favoriser une meilleure précision dans les termes d'optimisation, les instructions algorithmiques et la description des méthodes heuristiques.

Les prompts sont construits dans la continuité du même fil de discussion que celui utilisé en phase 1. Ainsi, lorsque les données de l'instance ont déjà été fournies ou parsées dans la conversation précédente, le modèle est invité à les réutiliser. Cette continuité permet de simuler une interaction progressive, dans laquelle le LLM est guidé à partir du contexte déjà établi.

3.5.1 Structure générale des prompts

Les prompts de phase 2 suivent une structure commune, quel que soit le problème étudié. Cette structure comprend :

1. **Une spécification du rôle** : le LLM est positionné comme expert en optimisation combinatoire et en méthodes heuristiques.
2. **Un rappel du problème** : le prompt indique que le modèle doit résoudre la même instance que celle traitée précédemment.
3. **Une heuristique imposée** : le prompt décrit les étapes exactes de la méthode à appliquer.
4. **Des contraintes expérimentales** : le modèle ne doit utiliser que les données de l'instance, sans recourir à des solutions optimales connues, des résultats publiés ou des méthodes exactes.
5. **Des tâches de résolution et de vérification** : le modèle doit produire une solution, calculer sa valeur objective et vérifier sa faisabilité.
6. **Un format de réponse final** : le prompt précise les informations attendues afin de faciliter l'extraction et la comparaison des résultats.

Cette structure permet de réduire les ambiguïtés et d'éviter les réponses trop générales. Elle oblige le modèle à suivre une démarche explicite et rend les réponses plus faciles à vérifier.

3.5.2 Contraintes imposées aux LLM

Afin de garantir une comparaison équitable avec la phase 1, plusieurs contraintes sont intégrées dans les prompts. Les modèles ne sont autorisés à utiliser que les données de l'instance fournie. Ils ne doivent pas rappeler, rechercher ou reproduire des solutions optimales connues, des meilleures solutions publiées ou des valeurs de référence associées aux benchmarks.

De plus, les méthodes exactes sont interdites dans cette phase. Les LLM ne doivent pas utiliser de solveur MILP, de *Branch-and-Bound*, de *Branch-and-Cut* ou toute autre méthode permettant de prouver directement l'optimalité. Pour le problème ULS, l'utilisation d'un algorithme exact de programmation dynamique est également interdite.

Les prompts demandent également au modèle de ne pas affirmer l'optimalité de la solution, sauf si celle-ci est réellement prouvée. Cette consigne est importante, car la première phase a montré que certains LLM pouvaient déclarer une solution optimale alors que la vérification montrait un écart, une infaisabilité ou une erreur de calcul.

3.5.3 Heuristiques guidées utilisées

Les heuristiques fournies aux LLM dépendent du problème traité. Elles ont été choisies à partir des méthodes observées dans la phase 1, en particulier celles associées aux meilleurs résultats obtenus par certains modèles. L’objectif est de tester si les modèles moins performants peuvent s’améliorer lorsqu’ils sont guidés par une méthode plus pertinente et explicitement structurée.

Dans cette phase, le choix des heuristiques ne vise pas à utiliser des méthodes exactes ou à garantir l’optimalité. Il s’agit plutôt de proposer des méthodes approchées classiques, compréhensibles et facilement implémentables, afin d’évaluer la capacité des LLM à suivre une procédure algorithmique, à respecter les contraintes du problème et à produire une solution vérifiable.

13

Tableau 3.1 : Heuristiques guidées utilisées dans la phase 2

Problème	Heuristique guidée fournie au LLM
p -médian	Prétraitement par plus courts chemins, construction gloutonne de type <i>Greedy ADD</i> , puis amélioration locale par recherche <i>Teitz–Bart 1-swap</i> .
Set Covering Problem	Construction gloutonne par efficacité coût/couverture, suppression des colonnes redondantes, puis amélioration locale par mouvements <i>add/drop/swap</i> .
Uncapacitated Lot Sizing	Initialisation par la méthode de Silver–Meal, amélioration par recherche locale, puis diversification par recuit simulé.

Heuristique guidée pour le problème du p -médian

Pour le problème du p -médian, l’heuristique guidée repose sur trois étapes principales : le prétraitement des distances, la construction d’une solution initiale par une méthode gloutonne, puis l’amélioration de cette solution par une recherche locale.

La première étape consiste à reconstruire la matrice complète des distances entre les nœuds. Dans les instances de type OR-Library, les données peuvent être fournies sous forme de graphe ou de matrice partielle. Il est alors nécessaire de calculer les plus courts chemins entre toutes les paires de nœuds afin d’obtenir une matrice de distances exploitable. Cette étape est importante, car le coût d’affectation d’un nœud à un centre dépend directement de la distance minimale entre ce nœud et le site sélectionné.

La deuxième étape est une construction gloutonne de type *Greedy ADD*. Cette méthode commence avec un ensemble vide de sites ouverts. À chaque itération, elle teste l’ajout de chaque site candidat non encore sélectionné et calcule la réduction du coût total d’affectation obtenue. Le site qui permet la plus grande diminution du coût est ajouté à la solution. Ce processus est répété jusqu’à obtenir exactement p sites ouverts.

L’intérêt de cette méthode est sa simplicité. Elle construit progressivement une solution en choisissant à chaque étape le site qui apporte le meilleur gain immédiat. Cependant, ce choix localement optimal ne garantit pas une solution globalement

optimale. Une fois qu'un site est sélectionné, la méthode ne revient pas automatiquement sur ce choix, ce qui peut conduire à une solution initiale de qualité moyenne.

La troisième étape vise donc à améliorer cette solution initiale par une recherche locale de type *Teitz–Bart 1-swap*. Le principe consiste à tester des échanges entre un site actuellement ouvert et un site non sélectionné. Pour chaque échange possible, le coût total d'affectation est recalculé. Si un échange permet de diminuer le coût, il est accepté. La procédure continue jusqu'à ce qu'aucun échange améliorant ne soit trouvé.

Cette recherche locale permet de corriger certaines limites de la construction gloutonne. Elle explore le voisinage de la solution en remplaçant progressivement certains sites par de meilleurs candidats. Dans le cadre de cette phase, l'objectif n'est pas de prouver l'optimalité, mais d'obtenir une solution faisable de meilleure qualité que celle produite par une simple construction gloutonne.

Pour ce problème, la faisabilité est principalement vérifiée à travers deux éléments : le nombre de sites sélectionnés doit être exactement égal à p , et chaque nœud de demande doit être affecté à l'un des sites ouverts. La valeur objective correspond ensuite à la somme des distances entre chaque nœud et le site ouvert auquel il est affecté.

Heuristique guidée pour le *Set Covering Problem*

Pour le *Set Covering Problem*, l'heuristique guidée repose également sur une logique en plusieurs étapes : construction gloutonne, suppression des redondances, puis recherche locale. Ce problème consiste à sélectionner un sous-ensemble de colonnes de coût minimal de manière à couvrir toutes les lignes de la matrice de contraintes.

La première étape est une construction gloutonne basée sur le rapport coût-efficacité. À chaque itération, l'algorithme considère les colonnes non encore sélectionnées et calcule, pour chacune d'elles, le rapport entre son coût et le nombre de lignes encore non couvertes qu'elle permet de couvrir. La colonne ayant le meilleur rapport, c'est-à-dire le coût le plus faible par nouvelle ligne couverte, est ajoutée à la solution.

Cette méthode est adaptée au SCP, car elle cherche à privilégier les colonnes qui couvrent beaucoup de contraintes encore insatisfaites pour un coût relativement faible. Elle permet généralement d'obtenir rapidement une solution faisable. Toutefois, comme toute méthode gloutonne, elle peut sélectionner certaines colonnes qui deviennent inutiles après l'ajout d'autres colonnes.

Pour cette raison, une deuxième étape de suppression des redondances est appliquée. Elle consiste à parcourir les colonnes sélectionnées et à vérifier si l'une d'elles peut être retirée sans rendre la solution infaisable. Une colonne est considérée comme redondante si toutes les lignes qu'elle couvre restent couvertes par d'autres colonnes sélectionnées. Dans ce cas, elle est supprimée afin de réduire le coût total.

Cette phase est importante, car la construction gloutonne tend parfois à produire des solutions faisables mais non minimales. La suppression des colonnes redondantes permet donc d'améliorer la solution sans ajouter de complexité excessive.

La troisième étape correspond à une recherche locale par mouvements de type *add/drop/swap*. Le mouvement *drop* consiste à retirer une colonne si la couverture reste complète. Le mouvement *add* consiste à ajouter temporairement une colonne afin de

permettre ensuite la suppression d'autres colonnes plus coûteuses. Le mouvement *swap* consiste à remplacer une ou plusieurs colonnes sélectionnées par une autre colonne, lorsque cela permet de conserver la couverture tout en diminuant le coût total.

Dans cette phase guidée, l'objectif est d'évaluer si le LLM est capable de respecter la logique de couverture complète. Pour le SCP, la faisabilité est particulièrement stricte : chaque ligne doit être couverte au moins une fois. Une seule ligne non couverte rend toute la solution infaisable, même si le coût total paraît faible. La vérification finale doit donc recalculer la couverture de chaque ligne à partir des colonnes sélectionnées.

Heuristique guidée pour l'*Uncapacitated Lot Sizing Problem*

Pour l'*Uncapacitated Lot Sizing Problem*, l'heuristique guidée est composée de trois étapes : une initialisation par la méthode de Silver-Meal, une amélioration par recherche locale, puis une diversification par recuit simulé.

Le problème ULS consiste à déterminer, sur un horizon de plusieurs périodes, les quantités à produire afin de satisfaire la demande tout en minimisant les coûts de lancement, de production et de stockage. Même si la capacité de production n'est pas limitée, le problème reste délicat, car les décisions prises à une période influencent les stocks et les coûts des périodes suivantes.

La première étape utilise l'heuristique de Silver-Meal. Cette méthode construit un plan de production en regroupant plusieurs demandes futures dans une même production. À partir d'une période donnée, l'algorithme compare le coût moyen par période lorsque l'on produit uniquement pour la période courante, puis lorsque l'on produit pour couvrir deux périodes, trois périodes, etc. Tant que le coût moyen diminue, le regroupement est considéré comme intéressant. Lorsque le coût moyen commence à augmenter, la méthode arrête le regroupement et lance une nouvelle production à la période suivante non couverte.

L'intérêt de Silver-Meal est de proposer un compromis entre deux stratégies extrêmes. Produire à chaque période réduit les coûts de stockage, mais augmente les coûts fixes de lancement. À l'inverse, produire une grande quantité à l'avance réduit le nombre de lancements, mais augmente les coûts de stockage. L'heuristique cherche donc à équilibrer ces deux effets en choisissant des lots de production raisonnables.

La deuxième étape consiste à appliquer une recherche locale sur la solution initiale. Elle peut modifier certains regroupements de périodes, déplacer un lancement de production, fusionner deux lots ou diviser un lot existant. À chaque modification, le plan est recalculé afin de vérifier que toutes les demandes sont satisfaites et que les stocks restent cohérents. Un mouvement est accepté s'il réduit le coût total tout en maintenant la faisabilité.

La troisième étape introduit une diversification par recuit simulé. Contrairement à une recherche locale classique, qui accepte uniquement les améliorations, le recuit simulé peut accepter temporairement une solution moins bonne avec une certaine probabilité. Cette probabilité diminue progressivement avec la température. L'objectif est d'éviter que l'algorithme reste bloqué dans un optimum local de mau-

vaise qualité.

Dans le cadre de cette phase, le recuit simulé permet donc d’explorer des plans de production plus variés. Par exemple, il peut accepter temporairement un regroupement plus coûteux si cela permet ensuite d’atteindre une meilleure combinaison deancements et de stocks. La qualité finale dépend alors de la manière dont les mouvements sont générés, de la température initiale, du taux de refroidissement et du nombre d’itérations.

Pour l’ULS, la vérification de faisabilité est plus délicate que pour les deux autres problèmes, car elle doit être réalisée période par période. Le stock final de chaque période doit respecter l’équation de bilan : le stock initial augmenté de la production doit couvrir la demande, et le stock restant est transféré à la période suivante. Une solution est infaisable si une demande n’est pas satisfaite, si un stock devient négatif ou si une production est déclarée sans décision de lancement cohérente lorsque des coûts fixes de setup sont utilisés.

Ainsi, l’heuristique guidée pour l’ULS permet d’évaluer la capacité des LLM à gérer une logique temporelle. Contrairement au p -médián et au SCP, où la solution peut être représentée par un ensemble de sites ou de colonnes, le Lot Sizing exige une cohérence séquentielle entre production, demande, stock et setup. Cette caractéristique rend le problème particulièrement utile pour tester la capacité des modèles à respecter des contraintes dynamiques.

3.5.4 Format de sortie demandé

Les prompts imposent un format de réponse final afin de faciliter la vérification automatique ou manuelle des résultats. Pour le problème du p -médián, le modèle doit notamment fournir :

- les indices des sites sélectionnés ;
- le nombre requis de sites p ;
- le nombre réel de sites sélectionnés ;
- l’affectation de chaque nœud de demande au site ouvert le plus proche ;
- le coût total d’affectation ;
- une vérification de faisabilité ;
- la méthode utilisée.

Pour le SCP, le modèle doit fournir les colonnes sélectionnées, le nombre de colonnes, le coût total et une vérification confirmant que toutes les lignes sont couvertes. Pour l’ULS, le modèle doit présenter un tableau par période indiquant la production, le stock et la décision de setup, ainsi que les différents coûts et la valeur objective totale.

Ce format imposé est important, car il limite les réponses vagues et permet de comparer les résultats aux indicateurs définis dans la phase 2. Il oblige également le LLM à expliciter les éléments nécessaires à la vérification : nombre de variables sélectionnées, cohérence des affectations, couverture des contraintes, calcul du coût et statut de faisabilité.

3.5.5 Informations de performance demandées

En plus de la solution finale, les prompts demandent des informations sur le déroulement de l’heuristique. Selon le problème, il peut s’agir du nombre d’itérations de construction gloutonne, du nombre de mouvements de recherche locale évalués, du nombre d’améliorations acceptées ou encore du temps de calcul algorithmique.

Les prompts demandent également de distinguer le **temps de génération de réponse** du **temps de calcul algorithmique**. Cette distinction est nécessaire car, dans le cas des LLM, le temps observé peut combiner plusieurs éléments : temps de raisonnement textuel, temps de génération du code, temps d’exécution éventuelle et temps de rédaction de la réponse finale.

Ainsi, la configuration des prompts de phase 2 permet de tester les LLM dans un cadre plus contrôlé que celui de la phase 1. Les modèles ne sont plus seulement évalués sur leur capacité à proposer spontanément une solution, mais sur leur capacité à appliquer une heuristique guidée, à respecter un format de sortie, à vérifier leur propre solution et à fournir des éléments permettant une comparaison rigoureuse.

3.6 Indicateurs d’évaluation de la phase 2

Afin d’évaluer l’effet du guidage, de nouveaux indicateurs sont introduits dans cette deuxième phase. Ces indicateurs permettent de mesurer non seulement la qualité finale de la solution, mais aussi la capacité du LLM à comprendre et implémenter correctement l’heuristique fournie.

Les indicateurs utilisés sont les suivants :

22

Tableau 3.2 : Indicateurs d’évaluation utilisés dans la phase 2

Indicateur	Description
Value of objective	Valeur objective obtenue par la solution proposée par le LLM.
Optimality of the value of objective	Indique si la valeur objective correspond à la valeur optimale ou à une valeur proche de l'optimum.
Time to generate the heuristic implementation	Temps nécessaire au LLM pour générer l'implémentation de l'heuristique demandée.
CPU LLM total time	Temps total associé à la génération, l'implémentation et/ou l'exécution de la solution proposée.
Correctness of objective value calculation	Vérifie si la valeur objective annoncée par le LLM correspond réellement à la solution fournie.
Capability	Mesure la capacité du LLM à produire une solution faisable et exploitable pour l'instance étudiée.
Reliability	Évalue la cohérence entre ce que le LLM affirme et ce que la vérification montre réellement.
Correctness of implementation	Vérifie si le code ou la méthode implémentée respecte correctement l'heuristique demandée.
Optimality GAP	Écart relatif entre la solution obtenue et la solution optimale ou de référence.
Improvement / Phase 1	Mesure l'amélioration ou la dégradation des résultats par rapport à la phase 1.

Ces indicateurs permettent de distinguer plusieurs niveaux de performance. Un LLM peut, par exemple, produire une solution faisable mais mal calculer sa valeur objective. Il peut aussi générer un code syntaxiquement correct mais qui n'applique pas réellement l'heuristique demandée. À l'inverse, un modèle peut ne pas atteindre l'optimum mais montrer une amélioration importante par rapport à la phase 1.

L'indicateur *Improvement / Phase 1* joue un rôle central. Il permet de mesurer l'effet réel du guidage : une réduction du gap, une amélioration de la faisabilité, une meilleure fiabilité ou une correction des erreurs observées précédemment.

3.7 Objectif de comparaison avec la phase 1

La comparaison avec la phase 1 constitue le cœur de cette deuxième expérimentation. En phase 1, les LLM étaient évalués comme des systèmes relativement autonomes : ils devaient comprendre le problème, choisir une méthode et produire une solution. En phase 2, ils sont placés dans un cadre plus guidé, où la méthode leur est fournie explicitement.

Cette comparaison permet d'analyser trois aspects principaux :

- **L'amélioration de la qualité des solutions** : le guidage permet-il de réduire le gap ou d'atteindre l'optimalité ?
- **L'amélioration de la faisabilité** : le guidage permet-il d'éviter les violations de contraintes observées en phase 1 ?
- **L'amélioration de la fiabilité** : le LLM annonce-t-il plus correctement le statut réel de sa solution ?

Cette analyse permet de mieux comprendre le rôle que peuvent jouer les LLM dans un processus de résolution. Si les résultats s’améliorent, cela suggère que les LLM peuvent être utiles comme assistants guidés, capables d’appliquer une méthode fournie par un expert. Si les résultats restent faibles, cela indique que le problème ne vient pas seulement du choix de la méthode, mais aussi de la capacité du modèle à raisonner, coder, calculer et vérifier correctement.

Ainsi, la phase 2 ne cherche pas seulement à obtenir de meilleures solutions. Elle cherche surtout à comprendre si le guidage transforme réellement le comportement des LLM face à un problème d’optimisation combinatoire.

3.8 Résultats obtenus sur le problème du p -médian

Le problème du p -médian constitue le premier cas d’application de la phase 2. Les résultats présentés dans cette section concernent uniquement les modèles qui avaient rencontré des difficultés en phase 1, c’est-à-dire les modèles ayant produit une solution infaisable ou une solution faisable avec un gap supérieur à 5%.

Au total, 43 tests guidés ont été réalisés sur les différentes instances du p -médian. Ces tests permettent d’évaluer si l’utilisation d’une heuristique imposée améliore les performances des LLM par rapport à la phase 1. L’analyse porte principalement sur la faisabilité des solutions, la qualité objective, la fiabilité, la correction de l’implémentation et l’amélioration par rapport aux résultats initiaux.

Les résultats présentés ci-dessous concernent uniquement les couples modèle-instance retestés en phase 2. Les couples ayant déjà obtenu une solution optimale ou quasi optimale en phase 1 ne sont donc pas inclus dans ce bilan.

12

Tableau 3.3 : Bilan global des résultats de la phase 2 sur le problème du p -médian

Indicateur	Résultat
Nombre total de tests guidés	43
Solutions faisables	29
Solutions infaisables	14
Solutions faisables avec un gap inférieur ou égal à 5	4
Réponses fiables	3
Implémentations correctes de l’heuristique	29
Améliorations par rapport à la phase 1	12

Ce bilan montre que le guidage heuristique améliore partiellement les résultats des LLM, notamment en termes de faisabilité et de correction de l’implémentation. En effet, 29 solutions sur 43 sont faisables, et le même nombre de cas présente une implémentation correcte de l’heuristique. Cependant, seulement 4 solutions faisables atteignent un gap inférieur ou égal à 5%, et seulement 3 réponses sont considérées comme fiables. Cela montre que le guidage aide les modèles à mieux appliquer une méthode, mais ne garantit pas nécessairement une solution proche de l’optimum ni une auto-évaluation correcte.

3.8.1 Analyse globale par modèle

Le tableau suivant présente une synthèse des résultats obtenus par chaque LLM dans la phase 2. Le gap moyen est calculé uniquement sur les cas pour lesquels une valeur numérique de gap est disponible, c'est-à-dire principalement les solutions faisables.

Tableau 3.4 : Synthèse des performances par LLM en phase 2 sur le p -médian

LLM	Tests	Faisables	Infaisables	Fiables	Implément. correcte	Amélior.	Gap moyen
ChatGPT 5.3	9	6	3	1	6	4	21.75
Gemini 3	13	8	5	0	8	1	35.96
DeepSeek	15	12	3	0	12	5	42.11
Grok 4	5	2	3	2	2	2	0.00
Claude Sonnet 4.6	1	1	0	0	1	0	2.71

Les résultats montrent que DeepSeek est le modèle le plus souvent retesté, avec 15 cas. Il produit 12 solutions faisables, ce qui indique une bonne capacité à générer des solutions respectant les contraintes après guidage. Cependant, son gap moyen reste élevé, avec 42.11%, et aucune réponse n'est considérée comme fiable. Le guidage permet donc à DeepSeek de produire davantage de solutions exploitables, mais sans garantir une bonne qualité objective.

Gemini 3 présente un comportement similaire mais légèrement moins favorable. Sur 13 tests, il produit 8 solutions faisables, mais une seule amélioration par rapport à la phase 1 est observée. Son gap moyen reste également élevé, avec 35.96%, et aucune réponse n'est fiable. Cela montre que le modèle applique parfois correctement l'heuristique, mais sans parvenir à améliorer significativement la qualité des solutions.

ChatGPT 5.3 obtient des résultats plus équilibrés. Sur 9 tests, il produit 6 solutions faisables, 6 implémentations correctes et 4 améliorations par rapport à la phase 1. Son gap moyen est de 21.75%, inférieur à celui de Gemini 3 et DeepSeek. Toutefois, une seule réponse est fiable, ce qui montre que le modèle peut améliorer ses solutions sans toujours annoncer correctement leur statut ou leur valeur objective.

Grok 4 présente le comportement le plus contrasté. Il n'est retesté que sur 5 cas, dont 3 restent infaisables. En revanche, les 2 solutions faisables obtenues sont optimales, fiables et issues d'une implémentation correcte. Ce modèle montre donc une forte performance lorsqu'il réussit à appliquer l'heuristique, mais son comportement reste instable.

Claude Sonnet 4.6 n'est retesté que sur une seule instance. Il obtient une solution faisable avec un gap faible de 2.71% et une implémentation correcte. Cependant, la réponse n'est pas fiable en raison d'une incohérence entre la valeur annoncée et la valeur recalculée. Comme un seul cas est disponible, il faut rester prudent dans l'interprétation de ses résultats.

3.8.2 Analyse de la faisabilité

La faisabilité constitue le premier critère d'évaluation, car une solution infaisable ne peut pas être exploitée, même si elle est bien présentée. Sur les 43 tests guidés, 29

solutions sont faisables, contre 14 solutions infaisables.

Ce résultat montre que le guidage heuristique améliore la capacité des LLM à produire des solutions respectant les contraintes du problème. La majorité des modèles parvient, dans plusieurs cas, à sélectionner un nombre correct de sites et à construire une solution compatible avec les contraintes du p -médian.

DeepSeek obtient le plus grand nombre de solutions faisables, avec 12 cas sur 15. Gemini 3 produit 8 solutions faisables sur 13, tandis que ChatGPT 5.3 en produit 6 sur 9. Ces résultats montrent que le guidage peut aider les modèles à mieux respecter les contraintes de base du problème.

Cependant, la faisabilité ne suffit pas à juger la qualité de la solution. Certaines solutions faisables restent très éloignées de l'optimum, avec des gaps importants. À l'inverse, Grok 4 produit seulement 2 solutions faisables sur 5, mais ces deux solutions sont optimales. Il est donc nécessaire de croiser la faisabilité avec le gap, la fiabilité et la correction de l'implémentation.

3.8.3 Analyse de la qualité objective et du gap

La qualité objective est évaluée à partir du gap entre la solution obtenue par le LLM et la solution optimale ou de référence. Sur les 43 tests guidés, seules 4 solutions faisables présentent un gap inférieur ou égal à 5%.

Ces cas sont les suivants :

- ChatGPT 5.3 sur `pmed17`, avec un gap de 0% après recalcul ;
- Grok 4 sur `pmed27`, avec un gap de 0% ;
- Claude Sonnet 4.6 sur `pmed30`, avec un gap de 2.71% ;
- Grok 4 sur `pmed35`, avec un gap de 0%.

Ces résultats montrent que le guidage peut permettre d'atteindre des solutions optimales ou proches de l'optimum, mais seulement dans un nombre limité de cas. La majorité des solutions faisables restent associées à des gaps élevés.

Le cas de Grok 4 est particulièrement significatif. Sur les instances `pmed27` et `pmed35`, il obtient des solutions optimales, avec une valeur objective correctement recalculée et une réponse fiable. Cependant, sur les autres cas testés, ses solutions restent infaisables. Le modèle présente donc une forte performance ponctuelle, mais une stabilité insuffisante.

ChatGPT 5.3 atteint également une solution optimale sur `pmed17`, mais la réponse reste non fiable à cause d'une incohérence entre la valeur annoncée et la valeur recalculée. Claude Sonnet 4.6 obtient une solution proche de l'optimum sur `pmed30`, mais présente le même problème de fiabilité.

Gemini 3 et DeepSeek produisent plusieurs solutions faisables, mais leurs gaps moyens restent élevés. Cela indique que le guidage améliore parfois la capacité à produire une solution valide, mais pas nécessairement la capacité à obtenir une solution compétitive.

3.8.4 Analyse de la fiabilité

La fiabilité est l'un des indicateurs les plus critiques dans cette phase. Sur les 43 tests guidés, seules 3 réponses sont considérées comme fiables. Ce résultat montre que même lorsque le LLM produit une solution faisable, il n'annonce pas toujours correctement son statut, sa valeur objective ou la qualité réelle de la solution.

Les réponses fiables sont observées principalement chez Grok 4, sur les instances pmed27 et pmed35, ainsi que chez ChatGPT 5.3 sur pmed40. Cependant, ces cas n'ont pas la même signification. Pour Grok 4, la fiabilité accompagne une solution optimale. Pour ChatGPT 5.3 sur pmed40, la réponse est fiable, mais le gap reste élevé, avec 41.34%. Cela montre qu'une réponse fiable n'est pas forcément une bonne solution : elle signifie seulement que ce que le modèle affirme est cohérent avec la vérification.

À l'inverse, certaines solutions de bonne qualité ne sont pas fiables. Par exemple, ChatGPT 5.3 obtient une solution optimale sur pmed17, mais la valeur objective annoncée n'est pas cohérente avec la valeur recalculée. De même, Claude Sonnet 4.6 obtient une solution faisable avec un gap de 2.71% sur pmed30, mais sa réponse n'est pas fiable en raison d'une incohérence dans le calcul ou l'annonce de la valeur objective.

Ces observations confirment que la fiabilité doit être analysée séparément de la faisabilité et du gap. Un LLM peut produire une bonne solution tout en donnant une mauvaise justification ou une mauvaise valeur. Pour un contexte d'aide à la décision, cette distinction est essentielle, même si elle est légèrement agaçante pour tous ceux qui espéraient une réponse magique en un clic.

3.8.5 Analyse de la correction de l'implémentation

La correction de l'implémentation permet d'évaluer si le LLM a réellement appliqué l'heuristique demandée. Sur les 43 tests, 29 implémentations sont considérées comme correctes. Ce résultat est nettement plus élevé que celui de la fiabilité, ce qui montre que les modèles peuvent suivre la méthode imposée sans pour autant évaluer correctement la qualité de leur propre solution.

ChatGPT 5.3 présente 6 implémentations correctes sur 9 tests. Gemini 3 en présente 8 sur 13, DeepSeek 12 sur 15, Grok 4 2 sur 5 et Claude Sonnet 4.6 1 sur 1. Ces résultats montrent que, dans de nombreux cas, les LLM parviennent à produire une implémentation conforme à l'heuristique guidée.

Cependant, une implémentation correcte ne garantit pas une solution de bonne qualité. Gemini 3 et DeepSeek présentent plusieurs implémentations correctes, mais leurs gaps moyens restent élevés. Cela indique que l'application de l'heuristique n'est pas suffisante pour atteindre des solutions proches de l'optimum. La qualité dépend aussi de la manière dont les choix gloutons, les affectations, les swaps et les calculs intermédiaires sont réalisés.

Cette analyse montre donc que la difficulté ne se limite pas à écrire ou suivre un algorithme. Elle concerne aussi la précision du calcul, la qualité des décisions prises à chaque étape et la capacité du modèle à vérifier correctement le résultat final.

3.8.6 Comparaison avec la phase 1 à travers l'indicateur Improvement / Phase 1

L'indicateur *Improvement / Phase 1* permet de comparer directement les résultats obtenus en phase 2 avec ceux de la phase 1 pour un même couple modèle-instance. Il permet d'évaluer si le guidage heuristique a réellement amélioré le comportement du LLM par rapport à son résultat autonome initial.

Une amélioration est interprétée de manière globale. Elle peut correspondre à une réduction du gap, au passage d'une solution infaisable à une solution faisable, à une meilleure correction du calcul objectif ou à une amélioration de la fiabilité.

Sur les 43 tests guidés réalisés, 12 montrent une amélioration par rapport à la phase 1.

Tableau 3.5 : Comparaison avec la phase 1 selon l'indicateur Improvement / Phase 1

LLM	Tests guidés	Améliorations	Interprétation
ChatGPT 5.3	9	4	Amélioration visible sur plusieurs instances, mais avec des problèmes persistants de fiabilité.
Gemini 3	13	1	Faible amélioration malgré plusieurs solutions faisables et implémentations correctes.
DeepSeek	15	5	Plusieurs solutions deviennent meilleures qu'en phase 1, mais les gaps restent élevés.
Grok 4	5	2	Amélioration très forte sur deux cas, avec deux solutions optimales et fiables, mais forte instabilité.
Claude Sonnet 4.6	1	0	Solution proche de l'optimum, mais sans amélioration notable selon l'indicateur retenu.

Les améliorations les plus significatives concernent notamment les cas où le guidage permet de réduire fortement le gap. ChatGPT 5.3 atteint par exemple une solution optimale sur pmed17, même si la réponse reste non fiable. Grok 4 obtient les améliorations les plus solides, avec des solutions optimales et fiables sur pmed27 et pmed35.

DeepSeek présente le plus grand nombre d'améliorations après ChatGPT 5.3 et Grok 4, avec 5 cas améliorés. Cependant, ces améliorations restent relatives, car les gaps demeurent élevés. Le guidage permet donc de progresser par rapport à la phase 1, mais pas toujours d'atteindre une solution de bonne qualité.

Gemini 3 montre une amélioration beaucoup plus limitée, avec un seul cas amélioré sur 13. Ce résultat suggère que le guidage heuristique ne suffit pas à corriger ses difficultés sur le problème du p -médian. Claude Sonnet 4.6, quant à lui, n'est retesté que sur un seul cas, ce qui ne permet pas de tirer une conclusion générale.

3.8.7 Synthèse des résultats sur le p -médian

Les résultats de la phase 2 sur le problème du p -médian montrent que le guidage heuristique a un effet positif, mais limité. Il améliore surtout la capacité des LLM à produire des solutions faisables et à appliquer une heuristique imposée. En revanche, il ne garantit pas une solution proche de l'optimum ni une réponse fiable.

La faisabilité s'améliore dans plusieurs cas, avec 29 solutions faisables sur 43. La correction de l'implémentation est également relativement élevée, avec 29 implémentations correctes. Cependant, seulement 4 solutions faisables présentent un gap inférieur ou égal à 5%, et seulement 3 réponses sont fiables. Ces résultats montrent que les LLM peuvent suivre une méthode guidée, mais restent limités dans l'évaluation correcte de leurs propres résultats.

Grok 4 obtient les meilleurs résultats ponctuels, avec deux solutions optimales et fiables. ChatGPT 5.3 montre une capacité d'amélioration intéressante, notamment sur `pmed17`, mais reste limité par des incohérences dans le calcul de la valeur objective. DeepSeek produit beaucoup de solutions faisables, mais avec des gaps élevés. Gemini 3 montre peu d'amélioration malgré plusieurs solutions faisables. Claude Sonnet 4.6 obtient une bonne solution sur l'unique cas retesté, mais sans fiabilité suffisante.

La conclusion principale est donc que les LLM peuvent bénéficier d'un guidage heuristique, mais seulement lorsque ce guidage est correctement compris, appliqué et vérifié. Ils ne peuvent pas être considérés comme des solveurs autonomes fiables pour le problème du p -médian. En revanche, ils peuvent jouer un rôle d'assistants guidés, à condition que leurs résultats soient systématiquement contrôlés par une vérification externe.

3.9 Résultats obtenus sur le *Set Covering Problem*

Le *Set Covering Problem* constitue le troisième problème étudié dans la phase 2. Comme pour les deux problèmes précédents, les résultats présentés dans cette section concernent uniquement les couples modèle-instance ayant rencontré des difficultés en phase 1, c'est-à-dire les cas où le modèle avait produit une solution infaisable ou une solution faisable avec un gap supérieur à 5%.

Dans cette phase guidée, les LLM reçoivent une heuristique structurée autour d'une construction gloutonne basée sur le rapport coût-efficacité, suivie d'une suppression des redondances et d'une recherche locale de type ajout, suppression et échange. L'objectif est d'évaluer si ce guidage permet d'améliorer la faisabilité, la qualité objective et la fiabilité des solutions générées.

Au total, 48 tests guidés ont été réalisés sur les instances du *Set Covering Problem*. Les résultats sont analysés selon les mêmes dimensions que précédemment : faisabilité, qualité de la solution, fiabilité, correction de l'implémentation et amélioration par rapport à la phase 1.

Les résultats présentés ci-dessous concernent uniquement les couples modèle-instance retestés en phase 2. Les cas déjà optimaux ou proches de l'optimum en phase 1 ne sont donc pas inclus dans ce bilan.

Tableau 3.6 : Bilan global des résultats de la phase 2 sur le *Set Covering Problem*

Indicateur	Résultat
Nombre total de tests guidés	48
Solutions faisables	9
Solutions infaisables	39
Solutions faisables avec un gap inférieur ou égal à 5%	8
Réponses fiables	
Implémentations correctes de l'heuristique	10
Améliorations par rapport à la phase 1	9

Ce bilan montre que le guidage heuristique reste limité sur le *Set Covering Problem* en termes de faisabilité. En effet, seules 9 solutions sur 48 sont faisables. Cependant, parmi ces solutions faisables, 8 atteignent un gap inférieur ou égal à 5%. Cela signifie que les LLM rencontrent encore des difficultés à produire des solutions valides, mais que lorsqu'ils y parviennent, la qualité objective peut être satisfaisante.

3.9.1 Analyse globale par modèle

Le tableau suivant présente une synthèse des résultats obtenus par chaque LLM dans la phase 2. Le gap moyen est calculé uniquement sur les solutions faisables disposant d'une valeur de gap numérique.

Tableau 3.7 : Synthèse des performances par LLM en phase 2 sur le *Set Covering Problem*

LLM	Tests	Faisables	Infaisables	Fiables	Implément. correcte	Amélior.	Gap moyen
ChatGPT 5.3	10	3	7	3	3	3	1.01
Gemini 3	15	1	14	1	1	1	0.00
DeepSeek	14	0	14	0	1	0	–
Grok 4	4	0	4	0	0	0	–
Claude Sonnet 4.6	5	5	0	4	5	5	2.89

Les résultats montrent que Claude Sonnet 4.6 obtient les meilleurs résultats sur les cas retestés du *Set Covering Problem*. Sur 5 tests guidés, il produit 5 solutions faisables, 5 implémentations correctes et 5 améliorations par rapport à la phase 1. Son gap moyen est de 2.89%, ce qui indique une bonne qualité objective sur les solutions obtenues.

ChatGPT 5.3 produit 3 solutions faisables sur 10 tests. Ces trois solutions sont fiables, correctement implémentées et améliorent les résultats de la phase 1. Son gap moyen est faible, avec 1.01%, ce qui montre que le modèle peut produire des solutions de bonne qualité lorsqu'il parvient à respecter les contraintes du problème.

Gemini 3 obtient une seule solution faisable sur 15 tests. Cette solution est optimale selon le gap renseigné, mais le nombre élevé de solutions infaisables montre que le modèle reste globalement peu robuste sur ce problème.

DeepSeek et Grok 4 ne produisent aucune solution faisable sur les cas retestés. DeepSeek présente une seule implémentation correcte, mais celle-ci ne conduit pas à

une solution faisable. Grok 4, quant à lui, reste infaisable sur les 4 cas retestés. Il faut toutefois rappeler que ces résultats concernent uniquement les cas problématiques de la phase 1 et non l'ensemble des tests initiaux.

3.9.2 Analyse de la faisabilité

La faisabilité est un indicateur essentiel pour le *Set Covering Problem*, car une solution doit couvrir toutes les contraintes ou toutes les lignes du problème. Une solution qui ne couvre pas l'ensemble des éléments requis est directement infaisable, même si sa valeur objective semble faible.

Sur les 48 tests guidés, seules 9 solutions sont faisables, contre 39 solutions infaisables. Ce résultat montre que le SCP reste difficile pour les LLM dans la phase guidée. La principale difficulté semble liée à la vérification complète de la couverture. Les modèles peuvent proposer un ensemble de colonnes peu coûteux, mais ne vérifient pas toujours que toutes les lignes sont réellement couvertes.

Claude Sonnet 4.6 est le seul modèle qui obtient un taux de faisabilité parfait sur les cas retestés, avec 5 solutions faisables sur 5. ChatGPT 5.3 obtient 3 solutions faisables sur 10, tandis que Gemini 3 n'en obtient qu'une seule sur 15. DeepSeek et Grok 4 ne produisent aucune solution faisable dans les cas retestés.

Ces résultats montrent que le guidage heuristique ne suffit pas toujours à garantir la faisabilité. Pour le SCP, le contrôle de couverture reste une étape critique, car une seule contrainte non couverte rend toute la solution invalide.

3.9.3 Analyse de la qualité objective et du gap

La qualité objective est évaluée à partir du gap entre la solution obtenue par le LLM et la solution optimale ou de référence. Sur les 48 tests guidés, 8 solutions faisables présentent un gap inférieur ou égal à 5%.

Ces cas sont les suivants :

- ChatGPT 5.3 sur scp41, avec un gap de 1.86% ;
- ChatGPT 5.3 sur scp45, avec un gap de 1.17% ;
- ChatGPT 5.3 sur scp65, avec un gap de 0% ;
- Gemini 3 sur scp65, avec un gap de 0% ;
- Claude Sonnet 4.6 sur scp51, avec un gap de 3.16% ;
- Claude Sonnet 4.6 sur scp61, avec un gap de 0% ;
- Claude Sonnet 4.6 sur scp62, avec un gap de 2.74% ;
- Claude Sonnet 4.6 sur scp64, avec un gap de 3.05%.

Ces résultats montrent que les solutions faisables obtenues sont généralement de bonne qualité. Contrairement au Lot Sizing, où plusieurs solutions faisables restent

très éloignées de l’optimum, le SCP présente peu de solutions faisables, mais avec des gaps souvent faibles.

Claude Sonnet 4.6 est le modèle le plus performant sur cet indicateur, avec plusieurs solutions proches de l’optimum. ChatGPT 5.3 obtient également de bons résultats sur les solutions faisables, avec un gap moyen de 1.01%. Gemini 3 atteint une solution optimale sur une instance, mais cette réussite reste isolée.

L’analyse du gap montre donc que la difficulté principale du SCP n’est pas seulement la qualité objective, mais surtout la capacité à produire une solution faisable. Une fois la couverture correctement assurée, les solutions obtenues peuvent être très proches de la référence.

3.9.4 Analyse de la fiabilité

La fiabilité permet de vérifier si les affirmations du LLM sont cohérentes avec la vérification externe. Sur les 48 tests guidés, seules 8 réponses sont considérées comme fiables.

ChatGPT 5.3 obtient 3 réponses fiables, correspondant à ses 3 solutions faisables. Gemini 3 obtient une seule réponse fiable, sur l’instance scp65. Claude Sonnet 4.6 obtient 4 réponses fiables sur 5 tests. En revanche, DeepSeek et Grok 4 ne présentent aucune réponse fiable sur les cas retestés.

Ces résultats confirment que la fiabilité reste un point faible des LLM. Dans plusieurs cas, les modèles proposent une solution ou une valeur objective, mais la vérification montre que les contraintes de couverture ne sont pas respectées. Cela est particulièrement problématique pour le SCP, car une solution peut sembler correcte numériquement tout en étant infaisable.

Il est donc nécessaire de distinguer la qualité apparente de la solution et sa validité réelle. Une réponse bien structurée ne garantit pas que toutes les contraintes sont couvertes. Encore une fois, le style ne remplace pas la vérification, tragédie moderne mais prévisible.

3.9.5 Analyse de la correction de l’implémentation

La correction de l’implémentation permet d’évaluer si le LLM a réellement appliqué l’heuristique imposée : construction gloutonne basée sur le coût-efficacité, suppression des redondances et recherche locale par ajout, suppression ou échange de colonnes.

Sur les 48 tests guidés, 10 implémentations sont considérées comme correctes. Parmi ces cas, 9 correspondent à des solutions faisables. Une implémentation correcte est également observée chez DeepSeek, mais elle ne conduit pas à une solution faisable. Claude Sonnet 4.6 obtient 5 implémentations correctes sur 5 tests. ChatGPT 5.3 en obtient 3 sur 10, Gemini 3 une seule sur 15, DeepSeek une seule sur 14 et Grok 4 aucune sur 4.

Ces résultats montrent que la capacité à suivre correctement l’heuristique reste très variable selon les modèles. Pour le SCP, l’implémentation correcte est fortement liée à la faisabilité, car l’heuristique doit non seulement sélectionner des colonnes, mais

aussi vérifier que toutes les contraintes sont effectivement couvertes.

Cependant, une implémentation correcte ne garantit pas automatiquement une solution optimale. Elle indique surtout que le modèle a respecté la structure de la méthode demandée. La qualité finale dépend ensuite des choix réalisés pendant la construction gloutonne, la suppression des redondances et les mouvements de recherche locale.

3.9.6 Comparaison avec la phase 1 à travers l'indicateur Improvement / Phase 1

L'indicateur *Improvement / Phase 1* permet de comparer les résultats de la phase 2 avec ceux de la phase 1 pour un même couple modèle-instance. Il permet d'évaluer si le guidage heuristique a réellement amélioré le comportement du LLM par rapport à son résultat autonome initial.

Sur les 48 tests guidés réalisés, 9 montrent une amélioration par rapport à la phase 1.

Tableau 3.8 : Comparaison avec la phase 1 selon l'indicateur Improvement / Phase 1 pour le *Set Covering Problem*

LLM	Tests guidés	Améliorations	Interprétation
ChatGPT 5.3	10	3	Amélioration nette sur les cas faisables, avec des gaps faibles, mais plusieurs cas restent infaisables.
Gemini 3	15	1	Une seule amélioration observée, malgré un grand nombre de tests guidés.
DeepSeek	14	0	Aucune amélioration observée sur les cas retestés.
Grok 4	4	0	Aucune amélioration observée sur les cas retestés.
Claude Sonnet 4.6	5	5	Amélioration sur tous les cas retestés, avec des solutions faisables et proches de l'optimum.

Les améliorations les plus significatives concernent Claude Sonnet 4.6, qui améliore les 5 cas retestés. ChatGPT 5.3 montre également une amélioration intéressante sur 3 instances, avec des gaps faibles. Gemini 3 présente une seule amélioration, tandis que DeepSeek et Grok 4 ne montrent aucune amélioration sur les cas guidés.

Ces résultats indiquent que le guidage heuristique peut être efficace pour certains modèles, mais qu'il ne produit pas un effet uniforme. Pour le SCP, le gain apporté par le guidage dépend fortement de la capacité du modèle à appliquer correctement la procédure de couverture et à vérifier la faisabilité finale.

3.9.7 Synthèse des résultats sur le *Set Covering Problem*

Les résultats de la phase 2 sur le *Set Covering Problem* montrent que le guidage heuristique a un effet positif, mais uniquement pour certains modèles. Globalement,

seules 9 solutions sur 48 sont faisables, ce qui montre que le problème reste difficile pour les LLM dans les cas retestés. La principale difficulté concerne la vérification complète de la couverture des contraintes.

Cependant, les solutions faisables obtenues sont généralement de bonne qualité. Parmi les 9 solutions faisables, 8 présentent un gap inférieur ou égal à 5%. Cela montre que lorsque les modèles parviennent à produire une solution valide, celle-ci peut être proche de l'optimum.

Claude Sonnet 4.6 est le modèle qui bénéficie le plus clairement du guidage sur le SCP. Il produit 5 solutions faisables sur 5 tests, 5 implémentations correctes et 5 améliorations par rapport à la phase 1. ChatGPT 5.3 obtient également des résultats intéressants, avec 3 solutions faisables et des gaps faibles. Gemini 3 ne réussit qu'un seul cas, tandis que DeepSeek et Grok 4 ne produisent aucune solution faisable sur les cas retestés.

La conclusion principale est donc que le SCP met en évidence une différence importante entre faisabilité et qualité objective. Le guidage ne garantit pas que les LLM produisent une solution faisable, mais lorsqu'une solution faisable est obtenue, elle peut être très compétitive. Comme pour les autres problèmes, ces résultats confirment la nécessité d'une vérification externe systématique des solutions générées par les LLM.

3.10 Résultats obtenus sur l'*Uncapacitated Lot Sizing Problem*

L'*Uncapacitated Lot Sizing Problem* constitue le deuxième problème étudié dans la phase 2. Comme pour le problème du p -médian, les résultats présentés dans cette section concernent uniquement les modèles ayant rencontré des difficultés en phase 1, c'est-à-dire les modèles ayant produit une solution infaisable ou une solution faisable avec un gap supérieur à 5%.

Dans cette phase guidée, les LLM reçoivent une heuristique composée de trois étapes : une initialisation par la méthode de Silver-Meal, une amélioration par recherche locale, puis une diversification par recuit simulé. L'objectif est d'évaluer si cette méthode guidée permet aux modèles d'améliorer leurs résultats par rapport à la phase 1.

Au total, 45 tests guidés ont été réalisés sur les instances de Lot Sizing. Les résultats sont analysés selon les mêmes dimensions que précédemment : faisabilité, qualité de la solution, fiabilité, correction de l'implémentation et amélioration par rapport à la phase 1.

Les résultats présentés ci-dessous concernent uniquement les couples modèle-instance retestés en phase 2. Les cas déjà optimaux ou proches de l'optimum en phase 1 ne sont donc pas inclus dans ce bilan.

Tableau 3.9 : Bilan global des résultats de la phase 2 sur le Lot Sizing

Indicateur	Résultat
Nombre total de tests guidés	45
Solutions faisables	16
Solutions infaisables	29
Solutions faisables avec un gap inférieur ou égal à 5	3
Réponses fiables	6
Implémentations correctes de l'heuristique	15
Améliorations par rapport à la phase 1	12

Ce bilan montre que le guidage heuristique produit des améliorations dans certains cas, mais que son effet reste limité. Sur 45 tests, seules 16 solutions sont faisables, ce qui signifie qu'une majorité des réponses reste inexploitable du point de vue des contraintes. De plus, seules 3 solutions faisables atteignent un gap inférieur ou égal à 5%. Le guidage aide donc certains modèles à progresser, mais il ne suffit pas à garantir une solution de bonne qualité.

3.10.1 Analyse globale par modèle

Le tableau suivant présente une synthèse des résultats obtenus par chaque LLM dans la phase 2. Le gap moyen est calculé uniquement sur les solutions faisables disposant d'une valeur de gap numérique.

Tableau 3.10 : Synthèse des performances par LLM en phase 2 sur le Lot Sizing

LLM	Tests	Faisables	Infaisables	Fiables	Implément. correcte	Amélior.	Gap moyen
ChatGPT 5.3	13	3	10	2	3	1	135.63
Gemini 3	10	2	8	0	2	0	17.83
DeepSeek	13	5	8	0	5	5	84.92
Grok 4	6	5	1	3	5	5	11.07
Claude Sonnet 4.6	3	1	2	1	0	1	0.45

Les résultats montrent que ChatGPT 5.3 est retesté sur 13 cas, mais ne produit que 3 solutions faisables. Il obtient une solution de très bonne qualité sur l'instance LLM(90, 5), avec un gap de 0.40%, mais ses autres solutions faisables restent beaucoup plus éloignées de l'optimum. Son gap moyen est donc fortement pénalisé par des cas avec des écarts très élevés.

Gemini 3 produit 2 solutions faisables sur 10 tests, mais aucune amélioration par rapport à la phase 1 n'est observée. Même lorsque ses solutions sont faisables, elles ne permettent pas d'améliorer les résultats initiaux. Le guidage semble donc peu efficace pour ce modèle sur le Lot Sizing.

DeepSeek obtient 5 solutions faisables sur 13 tests et présente également 5 améliorations par rapport à la phase 1. Cependant, ses gaps restent très élevés, avec une moyenne de 84.92%. Cela montre que le guidage lui permet parfois de passer d'une solution infaisable à une solution faisable, mais sans atteindre une qualité proche de l'optimum.

Grok 4 présente les résultats les plus intéressants. Sur 6 tests, il produit 5 solutions faisables, dont plusieurs avec des gaps modérés. Il obtient également 5 améliorations par rapport à la phase 1 et 3 réponses fiables. Contrairement au p-médian où son comportement était très instable, il semble ici mieux exploiter l’heuristique guidée.

Claude Sonnet 4.6 n’est retesté que sur 3 cas. Il produit une seule solution faisable, mais celle-ci est proche de l’optimum, avec un gap de 0.45%. Toutefois, comme le nombre de tests est faible, il faut rester prudent dans l’interprétation de ses performances. Oui, une seule bonne réponse ne fait pas encore une carrière de solveur.

3.10.2 Analyse de la faisabilité

La faisabilité est un critère central pour le Lot Sizing, car une solution doit respecter les équations de bilan de stock, satisfaire toutes les demandes, éviter les stocks négatifs et garantir que toute production est associée à une décision de setup lorsque cette variable est utilisée.

Sur les 45 tests guidés, seules 16 solutions sont faisables, contre 29 solutions infaisables. Ce résultat montre que le Lot Sizing reste difficile pour les LLM, même lorsqu’une heuristique leur est fournie explicitement.

Grok 4 obtient le meilleur taux de faisabilité, avec 5 solutions faisables sur 6 tests. DeepSeek produit 5 solutions faisables sur 13, tandis que ChatGPT 5.3 n’en produit que 3 sur 13. Gemini 3 produit 2 solutions faisables sur 10, et Claude Sonnet 4.6 une seule solution faisable sur 3.

Ces résultats indiquent que la difficulté principale ne vient pas uniquement du choix de la méthode, mais aussi de la capacité du modèle à respecter les contraintes temporelles du Lot Sizing. En effet, contrairement au p-médian, le Lot Sizing impose une cohérence période par période entre production, demande et stock. Une petite erreur dans le bilan peut rendre toute la solution infaisable. Les LLM, visiblement, trouvent encore le stock plus mystérieux qu’il ne devrait l’être.

3.10.3 Analyse de la qualité objective et du gap

La qualité objective est évaluée à partir du gap entre la solution obtenue par le LLM et la solution optimale ou de référence. Sur les 45 tests guidés, seules 3 solutions faisables présentent un gap inférieur ou égal à 5%.

Ces cas sont les suivants :

- ChatGPT 5.3 sur LLM(90, 5), avec un gap de 0.40% ;
- Grok 4 sur LLM(90, 3), avec un gap de 2.11% ;
- Claude Sonnet 4.6 sur LLM(90, 5), avec un gap de 0.45%.

Ces résultats montrent que le guidage peut permettre d’obtenir des solutions proches de l’optimum, mais uniquement dans un nombre réduit de cas. La majorité des solutions faisables restent associées à des gaps élevés.

Le cas de ChatGPT 5.3 sur LLM(90, 5) est particulièrement intéressant : le modèle passe d'un gap très élevé en phase 1 à une solution presque optimale en phase 2. Cependant, cette amélioration reste isolée, car les autres solutions faisables de ChatGPT 5.3 présentent des gaps très importants.

Grok 4 est le modèle le plus régulier parmi les solutions faisables. Même lorsque ses gaps restent supérieurs à 5%, ils sont généralement plus faibles que ceux de DeepSeek ou ChatGPT 5.3. Par exemple, il obtient un gap de 7.63% sur LLM(120, 10) et un gap de 15.71% sur LLM(120, 8), ce qui montre une amélioration notable par rapport à la phase 1.

DeepSeek produit plusieurs solutions faisables, mais avec des gaps très élevés, souvent supérieurs à 80%. Le guidage améliore donc parfois la faisabilité, mais pas suffisamment la qualité objective. Gemini 3, quant à lui, obtient peu de solutions faisables et ne montre pas d'amélioration claire.

3.10.4 Analyse de la fiabilité

La fiabilité permet de vérifier si les affirmations du LLM sont cohérentes avec la vérification réelle de la solution. Sur les 45 tests guidés, seules 6 réponses sont considérées comme fiables.

Les réponses fiables sont observées principalement chez Grok 4, ChatGPT 5.3 et Claude Sonnet 4.6. Grok 4 obtient 3 réponses fiables, notamment sur les instances LLM(90, 1), LLM(90, 3) et LLM(120, 10). ChatGPT 5.3 obtient 2 réponses fiables, notamment sur LLM(90, 5) et LLM(120, 6). Claude Sonnet 4.6 obtient une réponse fiable sur LLM(90, 5).

Cependant, la fiabilité ne signifie pas nécessairement que la solution est proche de l'optimum. Par exemple, ChatGPT 5.3 est fiable sur LLM(120, 6), mais le gap reste très élevé, avec 79.88%. De même, Grok 4 est fiable sur LLM(90, 1), mais le gap est de 15.06%. Une réponse fiable signifie donc surtout que le modèle annonce correctement ce qu'il a produit, pas qu'il produit une solution excellente.

À l'inverse, certaines solutions proches de l'optimum restent rares. La fiabilité doit donc être croisée avec le gap et la faisabilité. Cette distinction est importante : un LLM peut être honnête sur une mauvaise solution, ou confiant sur une solution fautive. Dans les deux cas, la vérification externe reste indispensable, parce qu'apparemment faire confiance à une phrase bien écrite n'est toujours pas une méthode scientifique.

3.10.5 Analyse de la correction de l'implémentation

La correction de l'implémentation indique si le LLM a réellement appliqué l'heuristique demandée : initialisation par Silver-Meal, amélioration locale et diversification par recuit simulé. Sur les 45 tests, 15 implémentations sont considérées comme correctes.

Grok 4 obtient le meilleur résultat sur cet indicateur, avec 5 implémentations correctes sur 6 tests. DeepSeek obtient également 5 implémentations correctes, mais ses solutions restent souvent éloignées de l'optimum. ChatGPT 5.3 obtient 3 implémentations correctes, Gemini 3 en obtient 2, tandis que Claude Sonnet 4.6 n'en obtient

aucune selon cet indicateur.

Ces résultats montrent qu’une implémentation correcte ne garantit pas automatiquement une bonne solution. DeepSeek, par exemple, applique correctement l’heuristique dans plusieurs cas, mais conserve des gaps très élevés. Cela peut s’expliquer par la manière dont les choix locaux sont effectués, par une mauvaise gestion des mouvements d’amélioration ou par une diversification insuffisante.

Le cas de Claude Sonnet 4.6 est particulier : il obtient une solution proche de l’optimum sur LLM(90, 5), mais l’implémentation n’est pas considérée comme correcte. Cela montre qu’un modèle peut produire un bon résultat numérique sans suivre fidèlement la méthode imposée. Pour une étude de guidage heuristique, cette distinction est importante, car l’objectif n’est pas seulement d’obtenir une bonne solution, mais aussi de vérifier si le modèle applique réellement la méthode demandée.

3.10.6 Comparaison avec la phase 1 à travers l’indicateur *Improvement / Phase 1*

L’indicateur *Improvement / Phase 1* permet de comparer directement les résultats obtenus en phase 2 avec ceux de la phase 1 pour un même couple modèle-instance. Il permet d’évaluer si le guidage heuristique a réellement amélioré le comportement du LLM par rapport à son résultat autonome initial.

Une amélioration est interprétée de manière globale. Elle peut correspondre à une réduction du gap, au passage d’une solution infaisable à une solution faisable, à une meilleure correction du calcul objectif ou à une amélioration de la fiabilité.

Sur les 45 tests guidés réalisés, 12 montrent une amélioration par rapport à la phase 1.

Tableau 3.11 : Comparaison avec la phase 1 selon l’indicateur *Improvement / Phase 1* pour le Lot Sizing

LLM	Tests guidés	Amélioration	Interprétation
ChatGPT 5.3	13	1	Amélioration très forte sur un cas, mais résultats globalement instables et souvent infaisables.
Gemini 3	10	0	Aucune amélioration observée malgré quelques solutions faisables.
DeepSeek	13	5	Plusieurs passages vers des solutions faisables, mais avec des gaps encore élevés.
Grok 4	6	5	Amélioration nette sur la majorité des cas retestés, avec une meilleure faisabilité et des gaps plus modérés.
Claude Sonnet 4.6	3	1	Une amélioration importante sur une instance, mais nombre de tests trop limité pour généraliser.

Les améliorations les plus significatives concernent principalement Grok 4. Le modèle s’améliore sur 5 des 6 cas retestés, ce qui montre que le guidage heuristique a un effet positif sur son comportement pour le Lot Sizing. En particulier, il passe de gaps très élevés en phase 1 à des gaps plus modérés en phase 2 sur plusieurs instances.

ChatGPT 5.3 présente une amélioration très nette sur LLM(90, 5), où il obtient une solution presque optimale avec un gap de 0.40%. Cependant, cette amélioration reste isolée. Sur les autres cas, les résultats sont souvent infaisables ou très éloignés de l’optimum.

DeepSeek montre également 5 améliorations, mais celles-ci doivent être interprétées avec prudence. Le modèle améliore souvent la faisabilité par rapport à la phase 1, mais ses gaps restent très élevés. L’amélioration est donc réelle, mais insuffisante pour obtenir des solutions compétitives.

Gemini 3 ne présente aucune amélioration selon cet indicateur. Cela suggère que le guidage par Silver-Meal, recherche locale et recuit simulé ne lui permet pas de corriger ses difficultés sur ce problème. Claude Sonnet 4.6 montre une amélioration sur une instance, mais le faible nombre de tests limite l’interprétation.

3.10.7 Synthèse des résultats sur le Lot Sizing

Les résultats de la phase 2 sur le Lot Sizing montrent que le guidage heuristique a un effet positif, mais très variable selon les modèles. Le problème reste difficile pour les LLM, en particulier à cause des contraintes temporelles liées au bilan de stock, aux demandes par période et aux décisions de setup.

Globalement, seulement 16 solutions sur 45 sont faisables. Parmi elles, seules 3 atteignent un gap inférieur ou égal à 5%. Cela montre que le guidage ne suffit pas à garantir une solution proche de l’optimum. En revanche, il permet parfois d’améliorer la faisabilité ou de réduire fortement le gap par rapport à la phase 1.

Grok 4 est le modèle qui bénéficie le plus clairement du guidage sur ce problème. Il produit 5 solutions faisables sur 6 tests, 5 implémentations correctes et 5 améliorations par rapport à la phase 1. ChatGPT 5.3 obtient une très bonne solution sur une instance, mais son comportement global reste instable. DeepSeek améliore plusieurs cas, mais les gaps restent trop élevés. Gemini 3 ne montre pas d’amélioration significative. Claude Sonnet 4.6 obtient une solution proche de l’optimum sur un cas, mais avec trop peu de tests pour conclure.

La conclusion principale est donc que les LLM peuvent bénéficier d’un guidage heuristique pour le Lot Sizing, mais cette amélioration dépend fortement du modèle. L’heuristique guidée aide certains modèles à mieux structurer leur résolution, mais la vérification externe reste indispensable pour contrôler la faisabilité, la valeur objective et la cohérence des résultats.

3.11 Conclusion

Ce chapitre a présenté la deuxième phase expérimentale de cette étude, consacrée à l’évaluation guidée des LLM sur trois problèmes d’optimisation combinatoire : le problème du p -médian, le *Set Covering Problem* et l’*Uncapacitated Lot Sizing Problem*. Contrairement à la première phase, dans laquelle les modèles étaient évalués de manière relativement autonome, cette phase a consisté à leur fournir une heuristique précise afin d’analyser leur capacité à comprendre, implémenter et appliquer une méthode de résolution imposée.

Les résultats obtenus montrent que le guidage heuristique peut améliorer partiellement les performances des LLM, mais que son effet reste fortement dépendant du problème étudié et du modèle utilisé. Pour le problème du p -médian, le guidage améliore surtout la faisabilité et la correction de l'implémentation, avec 29 solutions faisables sur 43 tests. Cependant, seules 4 solutions atteignent un gap inférieur ou égal à 5%, ce qui montre que la faisabilité ne garantit pas nécessairement une bonne qualité objective.

Pour le *Set Covering Problem*, les résultats mettent en évidence une situation différente. Le nombre de solutions faisables reste faible, avec seulement 9 solutions faisables sur 48 tests. Toutefois, parmi ces solutions faisables, 8 présentent un gap inférieur ou égal à 5%. Cela montre que la difficulté principale du SCP réside surtout dans la production d'une solution valide couvrant toutes les contraintes. Lorsque cette faisabilité est atteinte, la solution obtenue peut être très proche de l'optimum.

Pour l'*Uncapacitated Lot Sizing Problem*, les résultats sont plus limités. Seules 16 solutions sur 45 sont faisables, et seulement 3 solutions atteignent un gap inférieur ou égal à 5%. Ce problème apparaît donc comme plus difficile pour les LLM, notamment à cause des contraintes temporelles liées au bilan de stock, à la satisfaction de la demande période par période et à la cohérence entre production et décisions de setup.

L'analyse par modèle montre également que le guidage ne produit pas un effet uniforme. Certains modèles, comme Grok 4 sur le Lot Sizing ou Claude Sonnet 4.6 sur le SCP, bénéficient fortement du guidage sur certains cas. D'autres modèles produisent plusieurs solutions faisables, mais avec des gaps élevés ou une fiabilité limitée. Ces résultats montrent qu'un LLM peut réussir à appliquer une heuristique sans nécessairement produire une solution proche de l'optimum, ou produire une solution correcte sans évaluer correctement sa propre réponse.

Un résultat important de cette phase concerne la distinction entre faisabilité, qualité objective, fiabilité et correction de l'implémentation. Ces dimensions ne sont pas équivalentes. Une solution peut être faisable mais éloignée de l'optimum, une réponse peut être fiable sans être performante, et une implémentation peut suivre correctement l'heuristique sans conduire à une bonne solution. Cette distinction confirme l'intérêt d'utiliser plusieurs indicateurs d'évaluation au lieu de se limiter à la seule valeur objective.

Ainsi, les résultats de la phase 2 confirment que les LLM ne peuvent pas être considérés comme des solveurs autonomes fiables pour les problèmes d'optimisation combinatoire étudiés. En revanche, ils peuvent jouer un rôle d'assistants guidés, capables de comprendre et d'appliquer certaines heuristiques dans des conditions contrôlées. Toutefois, leurs résultats doivent toujours être soumis à une vérification externe afin de contrôler la faisabilité, le calcul de la valeur objective, le respect de l'heuristique et la cohérence des conclusions fournies.

Ce chapitre montre donc que le guidage méthodologique constitue une piste intéressante pour améliorer l'utilisation des LLM en optimisation combinatoire, mais qu'il ne suffit pas à éliminer les erreurs de raisonnement, de calcul ou de vérification. Ces observations serviront de base à l'analyse comparative finale et à la discussion sur les limites et les perspectives d'utilisation des LLM comme outils d'aide à la résolution en recherche opérationnelle.

Conclusion

Ce mémoire s’est intéressé à l’évaluation des grands modèles de langage comme outils d’assistance à la résolution de problèmes d’optimisation combinatoire. Dans un contexte où l’intelligence artificielle générative est de plus en plus utilisée pour la rédaction, la programmation et l’aide à la décision, ce travail a cherché à analyser de manière rigoureuse la capacité des LLM à traiter des problèmes relevant de la recherche opérationnelle et du génie industriel.

L’objectif principal n’était pas de considérer les LLM comme des solveurs classiques, mais d’étudier leur comportement face à des problèmes structurés, pour lesquels les solutions doivent être faisables, calculables et vérifiables. Cette distinction est essentielle, car une réponse générée par un LLM peut être cohérente en apparence, tout en contenant des erreurs de raisonnement, de calcul ou de faisabilité.

La démarche adoptée repose sur deux phases expérimentales. Dans la première phase, les LLM ont été évalués de manière relativement autonome : ils devaient comprendre les instances, proposer une méthode de résolution, produire une solution et justifier leur démarche. Cette phase a permis d’observer leur comportement spontané, mais aussi leurs limites, notamment les solutions infaisables, les erreurs de calcul et les affirmations non vérifiées.

La deuxième phase a introduit un guidage heuristique. Pour le problème du p -médian, le guidage reposait sur un prétraitement par plus courts chemins, une construction gloutonne *Greedy ADD* et une amélioration locale *Teitz–Bart 1-swap*. Pour le *Set Covering Problem*, l’heuristique combinait une construction gloutonne basée sur le rapport coût-efficacité, une suppression des redondances et une recherche locale. Pour l’*Uncapacitated Lot Sizing Problem*, le guidage reposait sur l’heuristique de Silver–Meal, une recherche locale et une diversification par recuit simulé.

Les résultats obtenus montrent que le guidage heuristique peut améliorer certains aspects des réponses générées par les LLM, notamment la faisabilité ou la correction de l’implémentation. Cependant, son effet reste variable selon les modèles et les problèmes étudiés. Pour le p -médian, le guidage améliore surtout la faisabilité, sans garantir systématiquement un faible gap. Pour le *Set Covering Problem*, les solutions faisables sont rares, mais souvent proches de l’optimum lorsqu’elles sont obtenues. Pour l’*Uncapacitated Lot Sizing Problem*, les résultats restent plus limités, principalement à cause des contraintes temporelles liées au bilan de stock, à la demande par période et aux décisions de setup.

Ce travail met ainsi en évidence plusieurs apports. Il propose d’abord un cadre expérimental permettant d’évaluer les LLM sur des problèmes d’optimisation vérifiables. Il montre ensuite l’intérêt de comparer une utilisation autonome et une utilisation guidée des modèles. Enfin, il souligne la nécessité de distinguer plusieurs dimensions de performance : faisabilité, valeur objective, gap, fiabilité, correction

de l'implémentation et amélioration par rapport aux résultats initiaux. Une solution peut en effet être faisable sans être proche de l'optimum, et une réponse peut être bien formulée sans être fiable.

Les résultats confirment également l'intérêt potentiel d'une hybridation entre intelligence artificielle générative et recherche opérationnelle. Les LLM peuvent aider à structurer un problème, proposer une formulation, générer du code ou appliquer une heuristique. Cependant, ils ne peuvent pas être considérés comme des solveurs autonomes fiables. Leurs réponses doivent être systématiquement contrôlées par des outils de vérification, des scripts de validation ou des méthodes classiques d'optimisation.

Ce travail présente toutefois certaines limites. Le nombre de modèles et d'instances reste limité, et les résultats dépendent fortement des prompts utilisés. Les performances des LLM peuvent aussi évoluer dans le temps, puisque ces modèles sont régulièrement mis à jour. De plus, l'étude porte sur des problèmes classiques et déterministes, alors que des problèmes industriels plus complexes, dynamiques ou incertains pourraient révéler d'autres difficultés.

Plusieurs perspectives peuvent être envisagées. Il serait d'abord intéressant d'étendre l'étude à un plus grand nombre d'instances et à des problèmes plus complexes, notamment des variantes du routage comme le *Vehicle Routing Problem with Delivery Options*. Une autre perspective consisterait à développer une boucle hybride dans laquelle le LLM génère une solution, un vérificateur externe contrôle la faisabilité et la valeur objective, puis le modèle reçoit un retour pour corriger sa réponse. Enfin, il serait pertinent d'étudier davantage les LLM comme assistants de modélisation plutôt que comme générateurs directs de solutions.

En conclusion, les LLM représentent des outils prometteurs pour assister certaines tâches liées à l'optimisation combinatoire, mais ils doivent être utilisés avec prudence. Leur valeur ne réside pas dans leur capacité à remplacer les solveurs ou les experts en recherche opérationnelle, mais dans leur potentiel à accompagner la modélisation, l'expérimentation et l'exploration de méthodes. Pour être exploitables dans un contexte scientifique ou industriel, leurs réponses doivent rester encadrées par une vérification externe, une traçabilité expérimentale et une analyse critique rigoureuse.

Prompts guidés utilisés dans la phase 2

Ce chapitre présente les prompts guidés utilisés dans la deuxième phase expérimentale. Contrairement aux prompts de la phase 1, ces prompts imposent une heuristique précise à appliquer. L'objectif est d'évaluer la capacité des LLM à comprendre une méthode de résolution, à l'implémenter correctement, à produire une solution faisable et à vérifier les résultats obtenus.

A.1 Prompt guidé du problème p -median

Cette section présente le prompt guidé utilisé pour le problème du p -median. Le guidage repose sur une heuristique en trois étapes : prétraitement par plus courts chemins, construction gloutonne *Greedy ADD* et amélioration locale par recherche *Teitz-Bart 1-swap*.

A.1.1 Prompt guidé

[style=promptstyle, caption=Prompt guidé utilisé pour le problème p -median] You are an expert in combinatorial optimization and heuristic methods.

Now, solve the same OR-Library p -median instance using the following guided heuristic method:

Multi-stage heuristic:

1. Shortest-path preprocessing
2. Greedy ADD construction
3. Teitz-Bart 1-swap local search improvement

Important constraints:

- You are allowed to use only the benchmark instance and its input data.
- You are NOT allowed to directly use, recall, or reproduce known optimal solutions, best-known solutions, or published results associated with this p -median instance.
- You must not use an exact MILP solver, branch-and-bound, branch-and-cut, or any published optimal value.

- The solution must be obtained by explicitly applying the guided heuristic to the provided data.
- If the instance data has already been parsed in the previous conversation, reuse the same parsed data.

Tasks:

1) Heuristic method description

Apply the following heuristic:

Stage 1: Shortest-path preprocessing

- Read the p-median instance data according to the OR-Library format.
- If the input data is provided as graph edges, reconstruct the all-pairs shortest-path distance matrix.
- Use Floyd-Warshall or repeated Dijkstra to compute the shortest-path distances between all nodes.
- Clearly report the values of n , m , and p if available.
- Explain how the distance matrix is obtained from the input graph.

Stage 2: Greedy ADD construction

- Start with an empty set of selected facilities.
- At each iteration, evaluate every non-selected node as a candidate facility.
- For each candidate facility, compute the total assignment cost obtained by assigning every demand node to its nearest selected facility.
- Select the facility that gives the largest reduction in the total assignment cost.
- Add the selected facility to the solution.
- Update the assignment of each demand node to its nearest selected facility.
- Continue until exactly p facilities are selected.

Stage 3: Teitz-Bart 1-swap local search improvement

Try to improve the solution using 1-swap moves:

- A 1-swap move consists of replacing one selected facility with one non-selected facility.
- For each candidate swap, reassign every demand node to its nearest selected facility.
- Recalculate the total assignment cost after the swap.
- Accept the swap only if it decreases the total assignment cost.
- Repeat until no improving 1-swap move is found.

2) Final solution

Provide:

- The final selected facility indices.
- The required number of facilities p .
- The number of selected facilities.
- The assignment of each demand node to its nearest selected facility.
- The total assignment cost.

- A feasibility check confirming that exactly p facilities are selected and that every demand node is assigned to one selected facility.

3) Verification

- Recalculate the total assignment cost from the selected facilities and assignments.
- Verify that the number of selected facilities is exactly p .
- Verify that all selected facility indices are valid and unique.
- Verify that every demand node is assigned to its nearest selected facility.
- If the solution is infeasible, clearly report the reason and identify the problem:
 - wrong number of selected facilities,
 - duplicate facilities,
 - invalid facility indices,
 - missing assignments,
 - incorrect assignment rule.
- Do not claim optimality unless it has been proven by the method.

4) Performance information

Report:

- Number of greedy ADD construction iterations.
- Number of evaluated 1-swap moves, if available.
- Number of accepted improving swaps.
- Algorithmic computation time if available.

5) Response time

At the end, report the response generation time separately from the algorithmic computation time.

Final answer format:

- Selected facilities: [...]
- Required number of facilities p : ...
- Number of selected facilities: ...
- Total assignment cost: ...
- Feasibility: feasible / infeasible
- Method used: Shortest-path preprocessing + Greedy ADD construction + Teitz-Bart 1-swap local search

A.2 Prompt guidé du *Set Covering Problem*

Cette section présente le prompt guidé utilisé pour le *Set Covering Problem*. Le guidage repose sur une heuristique en trois étapes : construction gloutonne basée sur le rapport coût-efficacité, suppression des redondances et amélioration locale par mouvements *add/drop/swap*.

A.2.1 Prompt guidé

[style=promptstyle, caption=Prompt guidé utilisé pour le Set Covering Problem]
You are an expert in combinatorial optimization and heuristic methods.

Now, solve the same OR-Library Set Covering Problem instance scp using the following guided heuristic method:

Multi-stage heuristic: 1. Greedy cost-effectiveness construction 2. Redundancy removal 3. Add/drop/swap local search improvement

Important constraints: - You are allowed to use only the benchmark instance and its input data. - You are NOT allowed to directly use, recall, or reproduce known optimal solutions, best-known solutions, or published results associated with scp. - You must not use an exact MILP solver, branch-and-bound, branch-and-cut, or any published optimal value. - The solution must be obtained by explicitly applying the guided heuristic to the provided data. - If the instance data has already been parsed in the previous conversation, reuse the same parsed data.

Tasks:

1) Heuristic method description

Apply the following heuristic:

Stage 1: Greedy cost-effectiveness construction - Start with all rows uncovered. - At each iteration, select the column that minimizes:

cost of column / number of newly covered rows

- Add the selected column to the solution. - Update the set of covered and uncovered rows. - Continue until all rows are covered.

Stage 2: Redundancy removal - Check each selected column. - Remove a selected column if all rows remain covered without it. - Prefer removing redundant columns with higher cost first. - Continue until no redundant column can be removed.

Stage 3: Local search improvement

Try to improve the solution using add/drop/swap moves: - Drop move: remove a selected column if feasibility is preserved. - Swap move: replace one selected column with one non-selected column if the solution remains feasible and the total cost decreases. - Add/drop move: temporarily add one column, then remove one or more redundant columns if this reduces the total cost. - Repeat until no improving move is found.

2) Final solution

Provide: - The final selected column indices. - The total cost of the selected columns. - The number of selected columns. - A feasibility check confirming that every row is covered at least once.

3) Verification

- Recalculate the total cost from the selected columns. - Verify that all rows are covered. - If some rows are not covered, clearly report the infeasibility and identify the uncovered rows. - Do not claim optimality unless it has been proven by the method.

4) Performance information

Report: - Number of greedy construction iterations. - Number of columns removed during redundancy removal. - Number of local search iterations or improving moves. - Algorithmic computation time if available.

5) Response time

At the end, report the response generation time separately from the algorithmic computation time.

Final answer format: - Selected columns: [...] - Number of selected columns: ... - Total cost: ... - Feasibility: feasible / infeasible - Method used: Greedy cost-effectiveness + redundancy removal + add/drop/swap local search

A.3 Prompt guidé de l'*Uncapacitated Lot Sizing Problem*

Cette section présente le prompt guidé utilisé pour l'*Uncapacitated Lot Sizing Problem*. Le guidage repose sur une heuristique en trois étapes : initialisation par la méthode de Silver-Meal, amélioration par recherche locale et diversification par recuit simulé.

A.3.1 Prompt guidé

[style=promptstyle, caption=Prompt guidé utilisé pour l'*Uncapacitated Lot Sizing Problem*] You are an expert in combinatorial optimization and heuristic methods.

Now, solve the same *Uncapacitated Lot Sizing Problem* instance using the following guided heuristic method:

Multi-stage heuristic:

1. Silver-Meal initialization
2. Local search improvement
3. Simulated annealing diversification

Important constraints:

- You are allowed to use only the benchmark instance and its input data.
- You are NOT allowed to directly use, recall, or reproduce known optimal solutions, best-known solutions, or published results associated with this *Uncapacitated Lot Sizing Problem* instance.
- You must not use an exact MILP solver, branch-and-bound, branch-and-cut, dynamic programming exact algorithm, or any published optimal value.
- The solution must be obtained by explicitly applying the guided heuristic to the provided data.
- If the instance data has already been parsed in the previous conversation, reuse the same parsed data.

Tasks:

- 1) Heuristic method description

Apply the following heuristic:

Stage 1: Silver-Meal initialization

- Read the Uncapacitated Lot Sizing Problem instance data.
- Identify the number of periods, demand for each period, setup costs, unit production costs if applicable, and holding costs.
- Start from the first period with positive demand.
- Use the Silver-Meal heuristic to decide how many future periods should be covered by a production lot.
- For each candidate production lot starting at period t , compute the average cost per period, including setup and holding costs.
- Extend the lot to cover additional future periods as long as the average cost does not increase.
- Once the best coverage length is selected, create the production lot and move to the next uncovered demand period.
- Continue until all periods' demands are covered.

Stage 2: Local search improvement

Try to improve the initial Silver-Meal solution using local search moves:

- Shift move: move part or all of a production lot from one period to another feasible production period.
- Merge move: merge two consecutive production lots if this reduces the total cost.
- Split move: split one production lot into two lots if this reduces holding or total cost.
- Reallocation move: change the periods covered by a production lot while keeping all demands satisfied.
- After each move, recalculate the total cost.
- Accept a move only if it decreases the total cost and keeps the solution feasible.
- Repeat until no improving local search move is found.

Stage 3: Simulated annealing diversification

Use simulated annealing to further diversify and improve the solution:

- Start from the locally improved solution.
- Generate neighboring solutions using shift, merge, split, or reallocation moves.
- If a neighbor improves the total cost, accept it.
- If a neighbor is worse, accept it with a probability depending on the temperature and the cost increase.
- Gradually reduce the temperature according to a cooling schedule.
- Continue until the stopping criterion is reached, such as a maximum number of iterations, a minimum temperature, or no improvement for several iterations.
- Keep track of the best feasible solution found during the search.

2) Final solution

Provide:

- The final production plan by period.

- The setup decision for each period.
- The inventory level at the end of each period.
- A table with the following columns: period, production quantity, setup decision, and ending inventory.
- The total setup cost.
- The total holding cost.
- The total production cost if applicable.
- The total objective value.
- A feasibility check confirming that all demands are satisfied and that inventory balance constraints are respected.

3) Verification

- Recalculate the total cost from the final production plan.
- Verify that demand is satisfied in every period.
- Verify that inventory levels are non-negative.
- Verify that production occurs only in periods where a setup is activated, if setup decisions are used.
- Verify the inventory balance for each period.
- If the solution is infeasible, clearly report the reason and identify the problem:
 - unmet demand,
 - negative inventory,
 - incorrect inventory balance,
 - production without setup,
 - incorrect total cost calculation.
- Do not claim optimality unless it has been proven by the method.

4) Performance information

Report:

- Number of Silver-Meal construction steps.
- Number of local search iterations or improving moves.
- Number of simulated annealing iterations.
- Number of accepted improving moves.
- Number of accepted non-improving moves, if available.
- Algorithmic computation time if available.

5) Response time

At the end, report the response generation time separately from the algorithmic computation time.

Final answer format:

- Production plan by period: [...]
- Setup decisions by period: [...]
- Ending inventory by period: [...]

- Production table:

Period	Production quantity	Setup decision	Ending inventory		
			1	...	...
			2	...	...

- Total setup cost: ...

- Total holding cost: ...

- Total production cost if applicable: ...

- Total objective value: ...

- Feasibility: feasible / infeasible

- Method used: Silver-Meal initialization + local search + simulated annealing diversification

Bibliographie

- [1] D. Altit. Diabetes (1a parte). *Correo Farmacéutico*, 2003.
- [2] Centrale Nantes. Laboratoire des sciences du numérique de nantes (LS2N). <https://research.ec-nantes.fr/fr/laboratoires-de-recherche/laboratoire-des-sciences-du-numerique-de-nantes-ls2n>, 2026. Consulté le 30 avril 2026.
- [3] Yitian Chen, Jingfan Xia, Siyu Shao, Dongdong Ge, and Yinyu Ye. Solver-informed RL: Grounding large language models for authentic optimization modeling, 2025. arXiv:2505.11792.
- [4] Ali Daud, Mobushira Khan, Sakher Ghanem, Sami Alesawi, Saud Yonbawi, Raed Alsini, Omar Alghushairy, Manal Linjawi, and Abdulrahman Ahmed Gharawi. Generative AI: A comprehensive overview of large language models for prompt engineering and applications. *arXiv preprint*, 2024. arXiv:2412.17690.
- [5] Direction Générale de la Recherche Scientifique et du Développement Technologique. Entités de recherche. Consulté en 2026.
- [6] Ashraf Elnashar, Jules White, and Douglas C. Schmidt. Enhancing structured data generation with GPT-4o: Evaluating prompt efficiency across prompt styles. *Frontiers in Computer Science*, 2024.
- [7] Shengyu Feng, Weiwei Sun, Shanda Li, Ameet Talwalkar, and Yiming Yang. A comprehensive evaluation of contemporary ML-based solvers for combinatorial optimization, 2025. arXiv:2505.16952.
- [8] Andrew Kean Gao. Prompt engineering for large language models: A brief guide with examples for non-technical readers. Stanford University, 2023. Unpublished manuscript.
- [9] Aurélien Géron. *Hands-On Machine Learning with Scikit-Learn, Keras & TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems*. O'Reilly Media, Sebastopol, CA, 2 edition, 2019.
- [10] Reza Ghanbarzadeh and Seyedali Mirjalili. A structured review of large language models in metaheuristic optimisation. *Decision Analytics Journal*, 15:100587, 2025.

- [11] Andre Hottung, Federico Berto, Chuanbo Hua, Nayeli Gast Zepeda, Daniel Wetzel, Michael Römer, Haoran Ye, Davide Zago, Michael Poli, Stefano Marsaroli, Jinkyoo Park, and Kevin Tierney. VRPAgent: LLM-driven discovery of heuristic operators for vehicle routing problems, 2025. arXiv:2510.07073.
- [12] S. S. Hussain, N. Oliver, and D. C. Klonoff. *Insulin Pumps and Continuous Glucose Monitoring Made Easy E-Book*. Made Easy. Elsevier Health Sciences, 2015.
- [13] IMT Atlantique. Création d’imt atlantique issue de la fusion de mines nantes et télécom bretagne. <https://www.imt-atlantique.fr/fr/actualites/paul-friedel-directeur-d-imt-atlantique-est-elue-la-prestigieuse-academie-d> 2019. Consulté le 30 avril 2026.
- [14] IMT Atlantique. MODELIS : des solutions innovantes pour la logistique, la production et les services. <https://www.imt-atlantique.fr/fr/actualites/modelis-logistique-innovantes>, 2023. Consulté le 30 avril 2026.
- [15] IMT Atlantique. Automation, production and computer sciences department (DAPI) – research. <https://www.imt-atlantique.fr/en/about/departments/dapi/research>, 2026. Consulté le 30 avril 2026.
- [16] IMT Atlantique. Campus de nantes. <https://www.imt-atlantique.fr/fr/campus/sites/nantes>, 2026. Consulté le 30 avril 2026.
- [17] IMT Atlantique. Ingénieur pour transformer l’avenir. <https://www.imt-atlantique.fr/fr/l-ecole>, 2026. Consulté le 30 avril 2026.
- [18] LinkedIn. Comparing major AI disciplines: AI vs ML vs DL vs GenAI vs RAG vs AI agents. Publication LinkedIn, 2026. Consulté le 8 mars 2026.
- [19] Ministère de l’Enseignement supérieur et de la Recherche. Doctorat. Consulté en 2026.
- [20] Ministère de l’Enseignement supérieur et de la Recherche. Unités de recherche : un cadre juridique clarifié, 2021. Consulté en 2026.
- [21] Sergio Morales, Robert Clarisó, and Jordi Cabot. Impromptu: A framework for model-driven prompt engineering. *arXiv preprint*, 2024. arXiv:2406.00213.
- [22] M. Z. Naser. A review of prompt engineering techniques for large language models. *arXiv preprint*, 2023. arXiv:2407.12994.
- [23] ROADEF. Société française de recherche opérationnelle et d’aide à la décision. Consulté en 2026.
- [24] ROADEF 2026. Roadef 2026. Consulté en 2026.
- [25] Krishna Ronanki, Simon Arvidsson, and Johan Axell. Prompt engineering guidelines for using large language models in requirements engineering. In *International Working Conference on Requirements Engineering: Foundation for Software Quality*. Springer, 2023.

- [26] Nguyen Gia Hien Vu, Ke Wang, and G. Gary Wang. Effective prompting with ChatGPT for problem formulation in engineering optimization. *Engineering Applications of Artificial Intelligence*, 2024.
- [27] Lilian Weng. Prompt engineering. Blog post, Lil'Log, 2023. Consulté le 10 avril 2026.
- [28] Jules White, Quchen Fu, Sam Hays, Michael Sandborn, Carlos Olea, Henry Gilbert, Ashraf Elnashar, Jesse Spencer-Smith, and Douglas C. Schmidt. A prompt pattern catalog to enhance prompt engineering with ChatGPT. *arXiv preprint*, 2023. arXiv:2302.11382.
- [29] World Health Organization. Definition, diagnosis and classification of diabetes mellitus and its complications: report of a WHO consultation. Part 1, diagnosis and classification of diabetes mellitus, 1999. WHO/NCD/NCS/99.2.
- [30] Bowen Zhang and Pengcheng Luo. OR-LLM-Agent: Automating modeling and solving of operations research optimization problem with reasoning large language model, 2025. arXiv:2503.10009.
- [31] Qin Zhou and Jiu-Bing Sheu. The use of generative artificial intelligence (GenAI) in operations research: review and future research agenda. *Journal of the Operational Research Society*, 2025.