PROSIT : Choix du meilleur solveur

FODIL Nel | RICCO Mathéo | GOUADFEL Rayan
17 juin 2024

Table des matières

I. Introduction 2
II. Analyse du contexte 2
III. Objectifs 3
IV. Problématique 3
V. Plan d'Action 4
VI. Définitions 5
Conclusion 8
Webographie 9

Table des figures

I. Introduction

Dans le cadre de la recherche opérationnelle, le choix du solveur adéquat est crucial pour aborder efficacement des problèmes complexes tels que les tournées de véhicules (Vehicle Routing Problem - VRP). Ce rapport explore divers solveurs pour déterminer lequel est le plus adapté à résoudre le VRP, en prenant en compte les contraintes et exigences spécifiques du problème. La tâche de sélectionner le meilleur solveur est impérative non seulement pour garantir l'exactitude des résultats mais aussi pour optimiser les performances en termes de temps de calcul et de gestion des ressources. Nous discuterons de la manière dont différents solveurs peuvent affecter les résultats et les processus décisionnels dans les applications réelles, soulignant l'importance de cette sélection dans le contexte de problèmes NP-complets.

II. Analyse du contexte

Le problème de tournées de véhicules est un casse-tête classique en logistique qui consiste à optimiser les itinéraires des flottes de véhicules afin de minimiser les coûts tout en satisfaisant diverses contraintes, telles que les fenêtres de temps, les capacités des véhicules, et les distances de parcours. Reconnu pour sa complexité NP-complète, le VRP ne permet pas une résolution rapide et exacte pour tous les cas envisageables par des algorithmes polynomiaux standards. Cette complexité impose un défi particulier pour les solveurs, nécessitant des approches sophistiquées pour obtenir des solutions exploitables dans des délais raisonnables.

Le choix du solveur est donc essentiel, non seulement pour la qualité des solutions générées mais aussi pour l'efficacité opérationnelle des entreprises qui dépendent de ces systèmes pour leur logistique quotidienne. La capacité d'un solveur à intégrer des contraintes de réalité commerciale et à fournir des résultats dans un temps acceptable définit son utilité dans un environnement commercial. La discussion sur le choix du solveur prend place dans un contexte plus large de l'optimisation des ressources et de la réduction des coûts, tout en augmentant la réactivité et la flexibilité des opérations logistiques.

III. Objectifs

- Respect des Contraintes des Variables Entières : Assurer que toutes les solutions générées respectent les contraintes de variables entières, évitant ainsi les solutions fractionnaires souvent produites par les solveurs de programmation linéaire standard.

- Gestion de la Complexité NP-Complete : Appliquer des techniques adaptées pour naviguer à travers la complexité NP-complète du problème, telles que les algorithmes de branch-and-bound, pour réduire efficacement l'espace de recherche tout en explorant les solutions potentielles.

- Implémentation de Méthodes Robustes : Utiliser des méthodes de relaxation continue et de séparation et évaluation pour développer des approches qui permettent de trouver des solutions proches de l'optimalité, même avec des variables de décision entières.

- Optimisation de l'Efficacité du Calcul : Développer un algorithme qui maximise l'efficacité du calcul, permettant de traiter des cas réels de taille importante sans exiger des ressources computationnelles excessives.

- Stratégie pour les Résultats Approximatifs : Mettre en œuvre une stratégie qui permet de retourner les meilleures solutions possibles dans un temps de calcul limité, acceptant que la solution idéale puisse parfois être hors de portée dans les contraintes temporelles données.

IV. Problématique

Étant donné la nature NP-complète du problème de tournées de véhicules, quel solveur est capable de fournir des solutions exactes et efficaces qui respectent la contrainte des variables entières, tout en étant pratiquement applicable à grande échelle ?

V. Plan d'Action

- Mise en Œuvre de Branch-and-Bound :
    - Développement : Concevoir et implémenter une méthode de séparation et d'évaluation (branch-and-bound) spécifiquement adaptée pour traiter les contraintes de variables entières, en veillant à ce que chaque solution intermédiaire respecte ces contraintes.
    - Optimisation : Intégrer des heuristiques pour sélectionner efficacement les branches à explorer en priorité, afin de réduire le nombre de nœuds évalués et accélérer la convergence vers la solution optimale.

- Application de la Relaxation Continue :
    - Formulation : Utiliser la relaxation linéaire des contraintes pour calculer des bornes supérieures et inférieures des solutions, fournissant ainsi un cadre pour évaluer la qualité des solutions trouvées par branch-and-bound.
    - Analyse : Examiner les écarts entre les solutions relaxées et entières pour identifier les contraintes critiques et ajuster la formulation du problème si nécessaire.

- Expérimentations et Ajustements des Paramètres :
    - Test sur Diverses Instances : Effectuer des tests systématiques de l'algorithme sur des ensembles de données de taille variable pour évaluer sa performance et sa scalabilité.
    - Tuning : Ajuster les paramètres de l'algorithme, tels que les critères d'arrêt et les heuristiques de branchement, pour optimiser la convergence et la qualité des solutions.

- Intégration d'Algorithmes Hybrides :
    - Combinaison de Méthodes : Explorer l'intégration de méta-heuristiques (telles que le recuit simulé ou l'algorithme génétique) avec les méthodes exactes pour traiter efficacement les instances de grande taille.
    - Adaptation Dynamique : Développer une approche adaptative qui permet à l'algorithme de basculer entre des méthodes exactes et des approches heuristiques en fonction de la complexité observée durant l'exécution.

VI. Définitions

Algorithmes de Complexité Exponentielle : Un algorithme de complexité exponentielle est un algorithme dont le temps d'exécution (ou le nombre de ressources nécessaires) croît exponentiellement en fonction de la taille de l'entrée. Formulé mathématiquement, si un problème a une taille d'entrée n, le temps d'exécution d'un algorithme exponentiel peut être exprimé sous la forme O(2^n) ou (c^n) où c est une constante plus grande que 1. Ces algorithmes deviennent très rapidement impraticables à mesure que n augmente, car le temps de calcul ou la quantité de ressources nécessaires augmentent de manière très rapide.

Pulp : Pulp est une bibliothèque de programmation en Python utilisée pour modéliser des problèmes de programmation linéaire (PL) et de programmation linéaire en nombres entiers (PLNE). Elle permet de définir des variables, des contraintes et des fonctions objectifs de manière intuitive et de résoudre ces problèmes en utilisant différents solveurs disponibles.

Solveur : Un solveur est un logiciel ou un algorithme conçu pour trouver des solutions à des problèmes mathématiques, notamment ceux formulés sous forme de systèmes d'équations, d'inégalités, ou de fonctions à optimiser. Les solveurs peuvent être utilisés pour divers types de problèmes, y compris la programmation linéaire, la programmation non linéaire, la programmation en nombres entiers, la programmation quadratique, etc.

Solveurs Spécifiques pour PLNE :

- **CBC (Coin-or branch and cut)** : Un solveur open-source de programmation linéaire en nombres entiers qui utilise la méthode de branch-and-cut pour résoudre des problèmes de grande envergure. Très utilisé dans la communauté académique et parfait pour les étudiants en recherche opérationnelle.
- **CPLEX** : Solveur commercial hautement performant pour la programmation linéaire et en nombres entiers. Il offre des options avancées pour le traitement des problèmes complexes et peut être une référence en matière de comparaison de performances.
- **Gurobi** : Autre solveur commercial qui est souvent loué pour sa rapidité et sa capacité à gérer de grands ensembles de données et des modèles complexes de programmation linéaire en nombres entiers.

Techniques de Linéarisation :

Linéarisation des Modèles Non-Linéaires : Technique qui transforme un problème de programmation non linéaire en un problème linéaire, afin qu'il soit solvable par des méthodes de programmation linéaire. Crucial pour les problèmes où les relations entre variables ne sont pas initialement linéaires.

Heuristiques et Métaheuristiques :

Heuristique de Construction : Approche qui construit une solution initiale réalisable, souvent utilisée comme point de départ pour des méthodes d'optimisation plus complexes. Pertinent pour initialiser les solutions dans les problèmes de tournées de véhicules.

Recuit Simulé, Algorithmes Génétiques : Exemples de métaheuristiques qui peuvent être utilisés pour améliorer les solutions initiales obtenues par des méthodes exactes ou pour trouver des solutions acceptables dans des cas où les méthodes exactes sont trop coûteuses en temps.

Optimisation Combinatoire :

Problèmes de Tournées de Véhicules (Vehicle Routing Problem - VRP) : Un cas classique d'optimisation combinatoire où l'objectif est de minimiser le coût total des routes prises par une flotte de véhicules pour livrer des produits ou services. Typiquement NP-difficile et souvent traité par des approches de programmation linéaire en nombres entiers et des heuristiques.

Relaxation et Découpage :

Relaxation Lagrangienne : Méthode qui consiste à relâcher certaines contraintes du modèle pour simplifier le problème initial. Les contraintes relaxées sont ensuite gérées dans la fonction objective à travers des pénalités, permettant de guider l'exploration de l'espace de solution vers des régions prometteuses.

Branch-and-Bound

Le **Branch-and-Bound** est une méthode algorithmique pour résoudre des problèmes d'optimisation nécessitant des solutions entières. Elle fonctionne en divisant le problème initial en sous-problèmes plus petits (branches). Chaque sous-problème est évalué en calculant une limite supérieure ou inférieure du meilleur résultat possible (bound). Si cette limite est moins favorable que la meilleure solution déjà trouvée, le sous-problème est élagué, c'est-à-dire ignoré pour la suite du processus. L'algorithme continue de diviser et évaluer les sous-problèmes restants jusqu'à ce que toutes les possibilités soient explorées ou élaguées, assurant ainsi la découverte de la solution optimale. Cette méthode est particulièrement utile pour réduire l'espace de recherche et gérer les contraintes de discrétisation dans les problèmes où les variables doivent prendre des valeurs entières.

Conclusion

En conclusion, ce projet nous a permis de plonger en profondeur dans les complexités de la programmation linéaire et de comprendre comment elle peut être adaptée pour traiter des problèmes d'optimisation avec des variables entières, comme le problème de tournées de véhicules. L'exploration des méthodes de branch-and-bound et de relaxation continue, en particulier, a offert des perspectives sur la manière de gérer efficacement les contraintes NP-complètes. Bien que les défis inhérents à ces problèmes soient significatifs, l'utilisation judicieuse de techniques mixtes et l'ajustement des paramètres algorithmiques peuvent grandement améliorer les performances et l'efficacité des solutions. Cette expérience renforce l'importance d'une sélection stratégique des outils et méthodes dans le domaine de la recherche opérationnelle pour répondre de manière optimale aux exigences spécifiques de chaque problème.

Webographie

https://towardsdatascience.com/a-comprehensive-study-of-mixed-integer-programming-with-jump-on-julia-part-1-8d47418324d4

https://www.youtube.com/watch?v=2zKCQ03JzOY&ab_channel=Recherche Op%C3%A9rationnelle

https://dept-info.labri.fr/~robson/RO/cours9.pdf

https://transp-or.epfl.ch/courses/RechOp/09-10/slides/PDF/17-BranchBound.pdf

https://webia.lip6.fr/~fouilhoux/MAOA/Files/MAOA_ROOC_impr.pdf