Exact solution methods for large-scale discrete p-facility location problems

Cristian Durán Mateluna

INSTITUT POLYTECHNIQUE DE PARIS

NNT : 2024IPPAE001

Thèse de doctorat

Exact solution methods for large-scale discrete p-facility location problems

Thèse de doctorat de l'Institut Polytechnique de Paris
préparée à l'Ecole Nationale Supérieure de Techniques Avancées

École doctorale n°574 Mathematiques Hadamard (EDMH)
Spécialité de doctorat: mathématiques appliquées

Thèse présentée et soutenue à Palaiseau, le 23 janvier 2024, par

Cristian Durán Mateluna

Composition du Jury :

Safia Kedad-Sidhoum Professeure, CNAM, France.	Présidente
Ivana Ljubic Professeure, ESSEC Business School, France.	Rapporteuse
Francisco Saldanha da Gama Professeur, University of Sheffield, England.	Rapporteur
Olivier Péton Professeur, IMT Atlantique, France.	Examinateur
Andrea Simonetto Professeur, UMA, ENSTA Paris, France.	Examinateur
Sourour Elloumi Professeure, UMA, ENSTA Paris, France.	Directrice de thèse
Zacharie Alès Enseignant-chercheur, UMA, ENSTA Paris, France.	Co-encadrant de thèse

Acknowledgments

This research and all associated expenses were funded by the National Agency for Research and Development of Chile - ANID (Scholarship Phd. Program 2019-72200492). I would also like to thank the UMA laboratory of ENSTA and the FMJH foundation that financed different conferences and PhD schools that allowed me to enrich my doctoral training and meet people from the international scientific community of Operations Research and Location Science.

I would like to thank my thesis supervisors Souroir Elloumi and Zacharie Alès who were the ones who trusted me and accompanied me for more than 4 years since the internship of the master in Operations Research. These years were a time of much learning. I thank them for their patience and pedagogy to be able to answer (re-explain) my questions and to give me their suggestions to advance in a research of excellence. Without a doubt, I hope to continue to grow and collaborate with you in any way we can. I also hope to see you in Chile.

I am grateful for the time of the reviewers and jury members who contributed to the current version of this thesis and who made my thesis defense very constructive for my professional development as a researcher. I would like to express my gratitude to Professors Miguel Alfaro and Oscar Vásquez of USACH, who introduced me to the field of research, recommended me to pursue my PhD and are looking forward to seeing me as a new colleague in Chile.

Thank you very much to everyone who was part of this important stage of my life. Happy to have met many students and professors at the UMA laboratory. In particular, I thank my Natalia Jorquera, with whom we have been able to work together and share the last two years. Thank you for your friendship and your help inside and outside the lab. I also thank Franco Quezada who was essential to approve my master degree and start my PhD, with whom we will remain colleagues and friends in Chile. I also thank the friendship of those who completed our Latin-Chilean community, Carolina Garcia, Nicolas Becerra and Natalia Ortiz, among others.

Doing this thesis has been the biggest challenge I have had as a student and professional, and being the last year even more so because it was the year I became a father for the first time. In this sense, nothing would have been possible if I had not had my wife Karla Vidal by my side, supporting me and above all taking care that our son grew up well and as happy as possible. Mateo and I love you infinitely.

Above all, I thank God, since my Christian faith is the one that has guided my values and my decisions, among which was to do this PhD in France.

Soli Deo gloria

Synthèse(en français)

Introduction

L'une des branches les plus importantes de la *recherche opérationnelle* et le sujet principal de cette thèse est l'étude des *problèmes de localisation*, en particulier les *problèmes de localisation discrète*. Ces problèmes de localisation sont présents dans une grande variété d'applications, telles que le clustering, le districting, le routage des transports, et la localisation d'installations telles que les entrepôts, les usines industrielles, les stations-service, les commissariats de police ou les centrales électriques. L'étude de ces problèmes contribue à améliorer l'allocation des ressources, à réduire les coûts et à améliorer les prestations de services dans diverses communautés et industries. De plus, l'inclusion de l'incertitude dans la modélisation de ces problèmes permet d'anticiper la prise de décision face à différents scénarios défavorables. C'est le cas dans le contexte de catastrophes naturelles telles que les tremblements de terre ou les ouragans, pour lesquelles ces problèmes peuvent aider à identifier les emplacements optimaux pour les abris temporaires, les centres médicaux et les points de distribution des fournitures essentielles.

La résolution des problèmes de localisation discrète peut se révéler difficile en raison de leur taille et de la complexité de calcul inhérente à leur structure combinatoire exponentielle, la plupart d'entre eux étant des problèmes NP-Difficiles. Bien qu'il existe plusieurs solveurs génériques, tels que CPLEX ou Gurobi, capables de résoudre bon nombre de ces problèmes, le temps de calcul devient rapidement trop grand à mesure que la taille de l'instance augmente. Dans ce contexte, plusieurs méthodes de résolution ont été développées pour faire face à ces limitations, mais la plupart d'entre elles sont des méthodes d'approximation. Ainsi, proposer de nouvelles méthodes exactes de résolution pour les instances à grande échelle est un défi permanent.

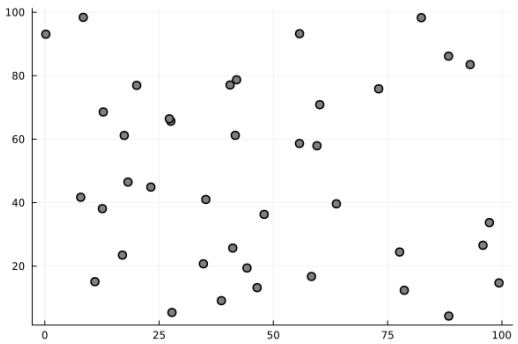
Problèmes de localisation discrète de p installations

Chaque problème de localisation comporte quatre éléments : (1) les clients, généralement déjà localisés, (2) les installations à localiser, (3) un espace où les installations et les clients sont situés, et (4) une métrique qui indique le coût, le temps ou la distance entre les installations et les clients (ReVelle and Eiselt (2005)). Parmi les problèmes fondamentaux de localisation discrète figurent le *Facility Location Problem (FLP)*, *l'Uncapacitated Facility Location Problem (UFLP)* et le *Capacitated Facility Location Problem (CFLP)* (Laporte et al. (2019b)). Le (FLP) consiste à sélectionner les emplacements optimaux d'un nombre prédéfini d'installations afin de satisfaire la demande des clients. L'objectif est de minimiser la somme des coûts de transport et de mise en

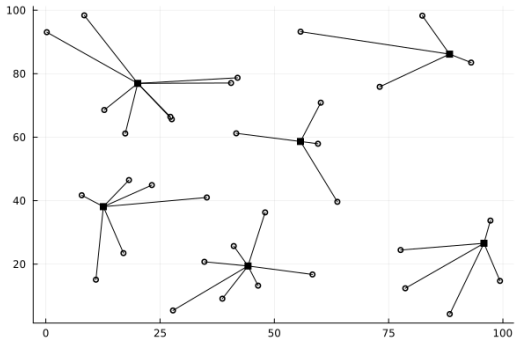
place. La $(CFLP)$ est une variante de la (FLP) dans laquelle les installations ont des capacités limitées pour satisfaire les clients. Enfin, dans le $(UFLP)$, ni les capacités des installations ni les demandes des clients ne sont prises en compte.

Cette recherche étudie les "*problems de p-installations*" dans lesquels il existe un nombre fixe de p installations à localiser, généralement sans tenir compte des coûts d'ouverture ou de fonctionnement. Nous exploitons cette structure particulière pour la solution exacte d'instances à grande échelle du problème *p-median (PMP)* et du problème *p-centre (PCP)*. Ces deux problèmes ne diffèrent que par leur fonction objective. Le (PMP) consiste à localiser p installations de telle sorte que la somme des distances d'allocation de tous les clients à leur installation la plus proche soit minimisée. En revanche, le (PCP) cherche à minimiser la plus grande distance d'allocation.

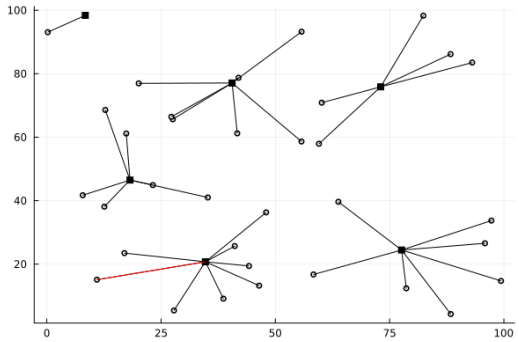
La Figure 1 illustre les solutions optimales de ces deux problèmes pour une instance avec 40 clients et sites candidats, et p égal à 6. Cette instance est dite symétrique puisque les sites candidats et les emplacements des clients sont identiques. Notez que dans la solution (PCP) (Figure 1c) une installation est allouée à seulement 2 clients isolés afin de réduire la distance maximale d'allocation (la ligne rouge) de 33 à 24. En revanche, la somme des distances d'allocation passe de 530 à 586. Le (PCP) est considéré comme plus équitable car il réduit la disparité des distances d'allocation. D'autre part, le (PMP) est considéré comme plus efficace puisque la distance moyenne d'allocation est minimisée.



(a) Instance symétrique avec 40 nœuds d'un carré de 100×100.



(b) Solution du p-median



(c) Solution du p-center

Figure 1: Comparaison des solutions optimales du (PMP) et du (PCP) pour la même instance (a) avec $p = 6$.

Portée et cadre théorique de la thèse

Dans cette thèse, nous proposons des modèles mathématiques et mettons en œuvre différentes méthodes de solutions exactes pour trois problèmes p-facilities. Le cadre théorique principal de cette recherche est l'optimisation combinatoire. Dans ce contexte, notre recherche se concentre sur l'étude de différentes formulations de problèmes de localisation discrète à l'aide de la programmation linéaire en nombres entiers mixtes (problèmes MILP).

Ce qui suit est une brève description des méthodes fondamentales de résolution des problèmes MILP, utilisées dans la littérature et que nous prenons également en compte.

- **Branch-and-Bound** ($B\&B$) : Partitionner l'espace des solutions réalisables en sous-espaces plus petits (branches d'un arbre de recherche) et résoudre la relaxation linéaire à chaque nœud. Des solutions réalisables en nombres entiers sont progressivement trouvées au fur et à mesure du parcours de l'arbre de recherche. ($B\&B$) incorpore des stratégies d'élagage pour éviter d'explorer des branches inutiles.

- **Méthodes de plans coupants** : Raffinement itératif de l'espace de solution à l'aide d'inégalités linéaires, appelées coupes.

- **Branch-and-Cut** ($B\&C$) : Une amélioration de l'approche ($B\&B$) qui incorpore des plans coupants pour renforcer les relaxations linéaires à chaque nœud.

- **Génération de colonnes** : Résoudre d'abord le problème avec seulement un sous-ensemble de ses variables. Puis, de manière itérative, les variables susceptibles d'améliorer la fonction objective sont ajoutées au programme jusqu'à ce qu'aucune autre amélioration ne soit possible.

- **Branch-and-Price** ($B\&P$) : Incorpore une génération de colonnes dans l'arbre de recherche branch-and-bound. À chaque nœud, la solution fractionnaire est évaluée pour déterminer si une nouvelle variable doit être ajoutée au problème.

- **Décomposition de Benders** : Divise le problème en un *problème principal* et un ou plusieurs *sous-problèmes*. Le problème principal est une relaxation du problème original, dans lequel certaines contraintes sont omises, ce qui le rend plus facile à résoudre. Les sous-problèmes tentent de fixer la valeur des variables omises en fonction de la solution du problème principal. Si cette solution n'est pas optimale, ils génèrent des coupes et le problème principal est à nouveau résolu.

Dans notre recherche, nous considérons également l'optimisation dans l'incertitude. Parmi les approches possibles, nous utilisons l'*optimisation robuste*. Cette approche considère que chaque paramètre incertain peut appartenir individuellement à un ensemble d'incertitude, puisqu'on ne dispose pas d'informations sur la distribution de probabilité. On vise à minimiser le coût du *pire cas*. En particulier, nous considérons une approche *en deux étapes* dans laquelle les décisions sont prises avant et après la révélation de l'incertitude.

Structure du manuscrit

La thèse est organisée comme suit. Tout d'abord, dans le Chapitre 2, nous présentons le cadre théorique de cette recherche avec les définitions et les méthodologies les plus importantes utilisées dans les autres chapitres. Dans le Chapitre 3, nous étudions une décomposition de Benders du problème *p-median*. Ce travail a été publié dans la revue *European Journal of Operational Research* (Duran-Mateluna et al. (2023a)). Le Chapitre 4 porte sur le problème *p-center* pour lequel la décomposition de Benders et une procédure de regroupement des clients sont étudiées. Dans le chapitre 5, nous étudions un problème robuste du *p-center* en deux étapes avec une incertitude sur les demandes et les distances des nœuds. Ce travail a été publié dans le numéro spécial de *Recent Advances in Location Science* de la revue *Computers & Operations Research* (Duran-Mateluna et al. (2023b)). Enfin, le Chapitre 6 présente une conclusion générale et examine certaines orientations pour la poursuite de la recherche.

L'introduction et les contributions avec les conclusions de chaque chapitre principal sont présentées ci-dessous.

Méthode de décomposition de Benders efficace pour le problème du p-median

Parmi les problèmes de localisation discrète, le problème *p-median* (PMP) est l'un des problèmes fondamentaux (Laporte et al. (2019b)). Dans le (PMP), les p sites doivent être choisis parmi l'ensemble des sites candidats sans tenir compte des coûts d'installation. Plus formellement, étant donné un ensemble de N clients, un ensemble de M centres potentiels à ouvrir, et leur ensemble d'indices correspondant $\mathcal{N} = \{1, ..., N\}$ et $\mathcal{M} = \{1, ..., M\}$, respectivement. Soit d_{ij} la distance entre le client $i \in \mathcal{N}$ et le site $j \in \mathcal{M}$ et $p \in \mathbb{N}$ le nombre de sites à ouvrir. L'objectif est de trouver un ensemble S de p sites tel que la somme des distances entre chaque client et son site le plus proche dans S soit minimisée.

La Figure 2 illustre une comparaison des solutions (PMP) pour différentes valeurs de p en considérant la même instance symétrique que celle utilisée dans la Figure 1.1a. On peut voir comment différents groupes sont identifiés. Au fur et à mesure que p augmente, chaque groupe comporte moins de nœuds et la distance moyenne d'allocation diminue par conséquent.

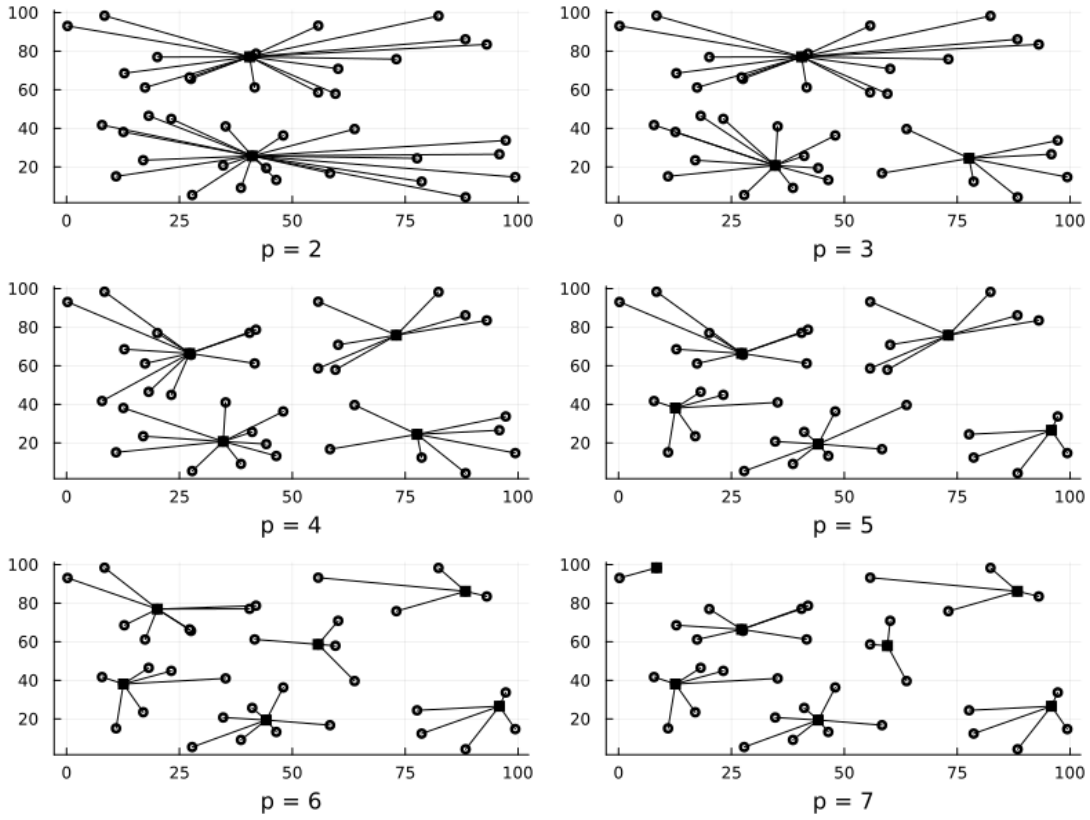


Figure 2: Comparaison des solutions optimales du (PMP) pour une instance symétrique de 40 nœuds.

Le (PMP) est un problème NP-hard (Kariv and Hakimi (1979)) et conduit à des applications où les sites correspondent à des entrepôts, des usines, des abris, etc. Cela inclut les contextes de la

logistique d'urgence et de l'aide humanitaire (An et al. (2014); Mu and Tong (2020); Takedomi et al. (2022)). Une autre application importante est un problème particulier de regroupement, généralement appelé *k-medoids problem* lorsque l'ensemble des clients et des sites sont identiques. Dans ce problème, des sous-groupes d'objets, de variables, de personnes, etc. sont identifiés en fonction de critères définis de proximité ou de similarité (Klastorin (1985); Park and Jun (2009); Marín and Pelegrín (2019); Ushakov and Vasilyev (2021); Voevodski (2021)).

Un grand intérêt pour la résolution des problèmes de localisation de grande taille a conduit au développement de diverses heuristiques et méta-heuristiques dans la littérature. Toutefois, la solution exacte des instances de grande taille reste un défi. Certains problèmes de localisation ont récemment été résolus efficacement à l'aide de la méthode de décomposition de Benders dans le cadre d'une approche *branch-and-cut* (voir par exemple Fischetti et al. (2017); Cordeau et al. (2019); Gaar and Sinnl (2022)). Parmi eux, le problème (UFL), dans lequel le nombre de sites à ouvrir n'est pas fixé, mais un coût d'ouverture est associé à chaque site. Par conséquent, nous explorons une décomposition de Benders pour le (PMP).

Contributions et conclusions

Une décomposition de Benders est effectuée sur la formulation la plus efficace du problème p-median. L'efficacité de la décomposition proposée provient d'un algorithme rapide pour la solution des sous-problèmes avec des améliorations dans la mise en œuvre d'une solution en deux étapes. Dans la première étape, les contraintes d'intégrité sont relâchées et dans la seconde étape, le problème est résolu par une approche efficace de type *branch-and-cut*. Notre approche est plus performante que les méthodes de l'état de l'art. Pour la première fois, des instances sont résolues avec jusqu'à 89600 et 238025 clients et sites des bibliothèques BIRCH et TSP, respectivement. L'algorithme de décomposition proposé est testé sur d'autres instances p-median : Les instances RW qui ne satisfont pas l'inégalité triangulaire et les instances ODM dans lesquelles il y a des allocations qui ne sont pas autorisées entre certains clients et sites. Pour les instances RW, il a été possible de résoudre des instances comptant jusqu'à 1000 clients avec une grande valeur de p. Pour les instances ODM avec 3773 clients, les instances précédemment non résolues ont été résolues en 10 heures. L'approche proposée a également été adaptée pour être testée sur les instances KG difficiles du problème (UFL), ce qui a permis d'obtenir des écarts d'optimalité relativement faibles.

Une des perspectives de cette recherche est d'exploiter ces résultats sur d'autres familles de problèmes de localisation. Il est également prévu d'utiliser d'autres stratégies de branchement qui permettent une plus grande efficacité lors du développement de l'algorithme *branch-and-cut*.

Algorithmes pour une nouvelle décomposition de Benders et un algorithme basé sur le clustering des clients pour le problème du p-centre.

Le problème du *p-centre* (PCP) est aussi un problème fondamental dans la science de la localisation Laporte et al. (2019b). Il consiste à choisir les p emplacements pour minimiser la distance maximale entre un client et l'établissement le plus proche. Les installations ouvertes p sont appelées *centres*. Formellement, le problème est défini comme suit. Étant donné un ensemble de N clients, un ensemble de M centres potentiels à ouvrir et leur ensemble d'indices correspondant $\mathcal{N} = \{1, ..., N\}$ et $\mathcal{M} = \{1, ..., M\}$, respectivement. Soit d_{ij} la distance entre le client i et le centre j. Soit p un entier. L'objectif est de trouver un ensemble $S \subseteq \mathcal{M}$ tel que $|S| \leq p$ et la valeur $z = \max_{i \in \mathcal{N}} \min_{j \in S} d_{ij}$ est minimisée. Cette distance z est appelée *rayon*. La Figure 3 illustre une comparaison des solutions (PCP) pour différentes valeurs de p en considérant la même instance symétrique que celle utilisée dans l'introduction ci-dessus. On peut noter que le rayon correspondant, indiqué en rouge, diminue à mesure que p augmente.

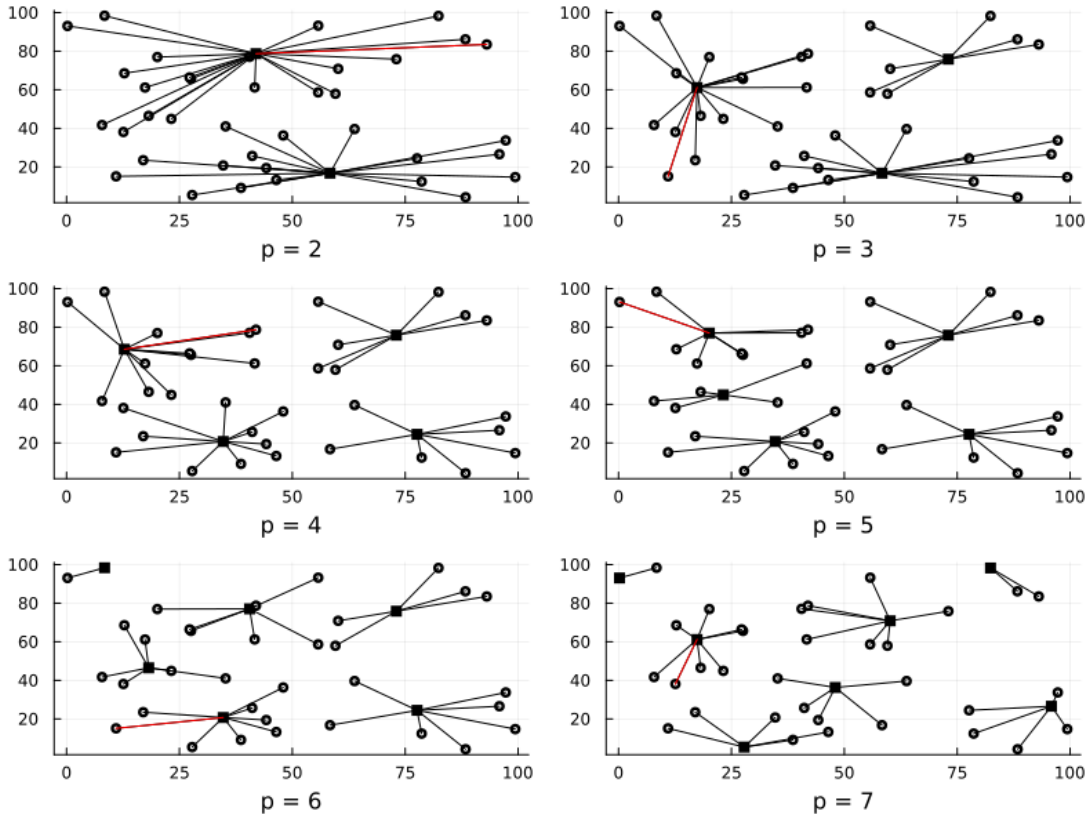


Figure 3: Comparaison des solutions du (PCP) optimales pour une instance symétrique de 40 nœuds.

Le problème du p-centre a des applications dans divers contextes, notamment la localisation des installations, la planification des services d'urgence, la conception des réseaux de télécommunications, la planification des soins de santé et la gestion de la chaîne d'approvisionnement. Il permet de décider des meilleurs emplacements pour les centres, tels que les entrepôts ou les hôpitaux, afin de minimiser les distances ou les temps de réponse (Çalık et al. (2019a)).

Il existe une variété de méthodes exactes et approximatives pour le problème classique et ses variantes. Ces dernières années, des progrès importants ont été réalisés dans la résolution d'instances à grande échelle. À notre connaissance, les meilleures méthodes exactes sont celles de Contardo et al. (2019) et de Gaar and Sinnl (2022). La première considère initialement un sous-ensemble de clients, résout le PCP associé à ce sous-ensemble, et ajoute de nouveaux clients jusqu'à l'obtention d'une solution optimale. La seconde méthode est basée sur une procédure spécialisée de *branch-and-cut* pour une décomposition de Benders. Les deux méthodes peuvent résoudre des instances à très grande échelle, car elles prennent en compte la propriété selon laquelle (PCP) peut être résolu en considérant un sous-ensemble de clients et en déterminant ensuite si la solution résultante est optimale pour les clients restants.

Contibution et conclusions

Il existe cinq formulations principales pour le (PCP), chacune ayant des performances différentes qui peuvent varier considérablement en fonction des paramètres du solveur, tels que le presolve. Parmi les formulations, $(F4)$ et $(F2)$ apparaissent être les meilleures options. Cependant, $(F2)$ est plus performant que $(F4)$ lorsque le presolve est désactivé, car il a moins de contraintes redondantes

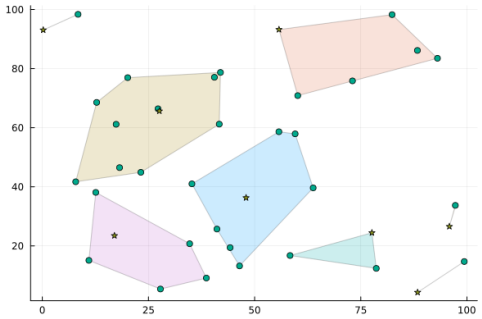
Deux décompositions de Benders ont été étudiées pour la formulation $(F2)$ en fonction de l'ensemble des variables à relaxer. Lorsque les variables de décision associées à la localisation des installations sont relâchées, nous obtenons une décomposition qui conduit à des problèmes de couverture d'ensemble comme sous-problème primaux et à des problèmes de cliques maximales comme sous-problèmes duaux. Cela montre une relation étroite avec l'une des méthodes classiques de solution exacte de (PCP), qui est la recherche binaire du rayon induisant la distance en résolvant les problèmes de recouvrement d'ensembles associés. Comme le problème maître est facile à résoudre, la décomposition de Benders peut être considérée comme une recherche linéaire de cette même distance, puisque pour chaque valeur de distance évaluée, nous vérifions s'il existe une solution réalisable de p sites. Notre implémentation permet d'obtenir des performances similaires pour les deux méthodes.

D'autre part, lorsque les variables associées à l'allocation des clients aux installations ouvertes sont relâchées, le sous-problème est facile à résoudre, ce qui permet une solution rapide. Nous montrons que la décomposition sur $(F2)$ permet d'avoir deux approches de séparation par une coupe unique, qui en variables continues conduisent à des valeurs différentes de relaxation linéaire pour le problème original. D'autre part, la relation entre cette décomposition de Benders $(F1)$ et l'approche multi-coupes présentée dans Gaar and Sinnl (2022) est également mise en évidence. Toutes les approches ont été testées, et bien que notre décomposition de Benders nous permette d'obtenir de meilleures bornes inférieures de relaxation linéaire, avec notre implémentation actuelle, elle prend plus de temps que l'approche multi-coupe.

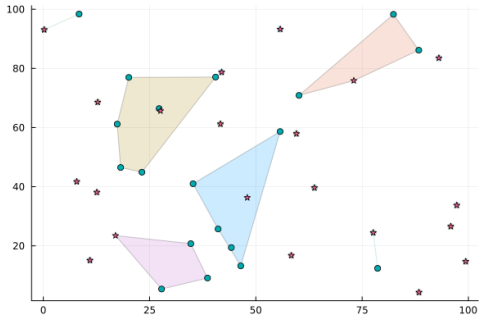
Enfin, une méthode exacte basée sur un regroupement des clients a été proposée, afin de résoudre le (PCP) itérativement pour l'ensemble des représentants associés et de les mettre à jour si nécessaire. Une première étape du problème (PCP) relaché a été considérée avant de passer à une seconde étape pour résoudre le (PCP) entier avec de bonnes bornes inférieures et supérieures. De plus, pour réduire le nombre de valeurs de distance différentes, nous considérons également une procédure itérative d'arrondis de la distance avec la fonction modulo. Toutes ces améliorations de l'implémentation permettent de dépasser l'état de l'art de Gaar and Sinnl (2022) et Contardo et al. (2019) sur les instances TSP de grande taille.

La Figure 4a montre la même instance de 40 nœuds précedente avec une répartition des clients en 8 groupes dans lesquels les représentants sont marqués par des étoiles. En appliquant notre algorithme, les clients sont retirés séquentiellement de ces groupes lorsque cela est nécessaire. La Figure 4b montre l'ensemble final des groupes. Par conséquent, avec l'ensemble final de représentants, nous pouvons trouver une solution optimale présentée dans la Figure 4c.

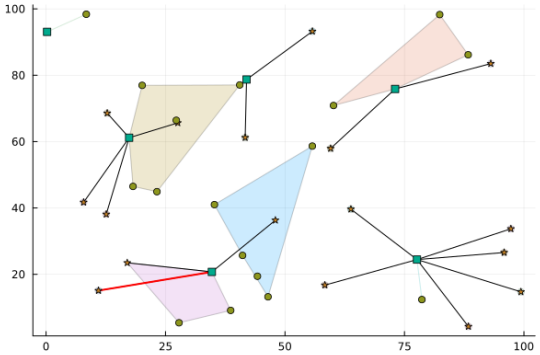
La perspective principale de la décomposition de Benders du (PCP) sur $(F2)$ est d'améliorer notre implémentation pour tirer parti de la force des coupes uniques en termes de borne inférieure, contrairement à l'approche multi-coupes. Parallèlement, les très bons résultats de l'algorithme de clustering suggèrent qu'il peut être appliqué à d'autres problèmes de localisation discrète ou à d'autres types d'instances.



(a) Ensemble initial de clusters



(b) Ensemble final de clusters



(c) Solution du 6-centre avec les clusters

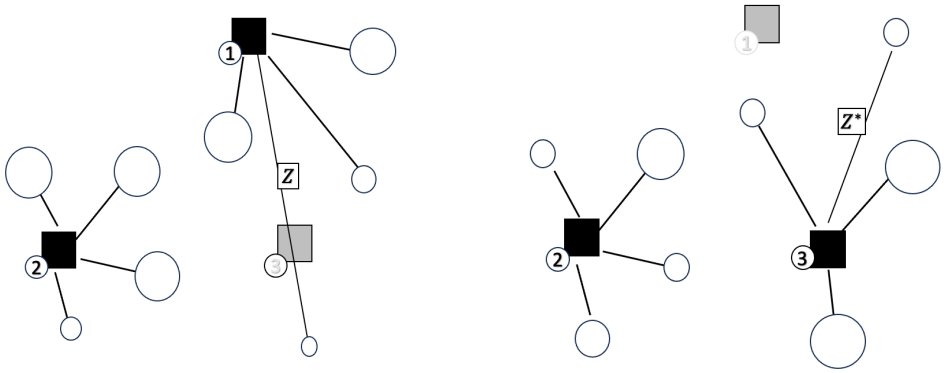
Figure 4: Illustration de la procédure de clustering pour le (PCP) sur l'instance de 40 nœuds avec $p = 6$

Formulations MILP et algorithmes exacts pour le problème robuste du p-centre en deux étapes

Le problème p-center (PCP) sous incertitude est bien étudié dans la littérature (Çalık et al. (2019b)). Ce problème se pose lorsque des paramètres, tels que les demandes ou les distances entre les nœuds de demande et les sites disponibles, varient dans le temps ou lorsque leur valeur exacte est incertaine. Dans l'optimisation robuste, l'incertitude est généralement représentée par des paramètres qui peuvent prendre n'importe quelle valeur dans un ensemble d'incertitude. Chaque réalisation de l'ensemble d'incertitude est appelée "scénario". Les ensembles les plus classiques sont la boîte, l'ellipsoïde et les ensembles d'incertitude budgétisés (voir, par exemple, Ben-Tal et al. (2009); Bertsimas and Sim (2004); Du and Zhou (2018); Paul and Wang (2019)).

Une fonction objective classique dans l'optimisation robuste est le regret que l'on a pour la décision initiale une fois que l'incertitude est révélée. La Figure 5 illustre comment la solution optimale d'un problème peut changer en fonction des valeurs de ses paramètres. Dans cet exemple, nous considérons un problème de p-centre dans lequel chaque client a une demande incertaine. La valeur de la demande est représentée par la taille des cercles. La Figure 5a 1a représente la valeur des paramètres initiaux et la solution optimale correspondante qui consiste à ouvrir les sites 1 et 2. Une fois l'incertitude révélée, les paramètres prennent la valeur représentée sur la Figure 5b.

On constate que cela modifie la solution optimale qui consiste désormais à ouvrir les sites 2 et 3. Dans ce contexte, le regret correspond à la différence entre les valeurs objectives des deux solutions lorsque les paramètres prennent les valeurs représentées dans la Figure 5b.



(a) La solution avant l'incertitude (b) La solution après l'incertitude

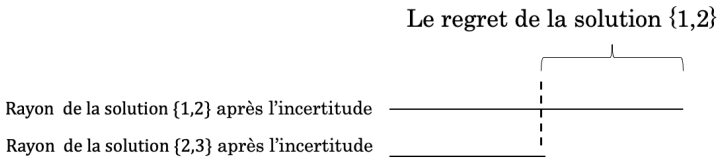


Figure 5: Exemple de regret des solutions optimales (RPCP) avant et après l'incertitude

L'incorporation de l'incertitude dans (PCP) a des applications importantes dans les problèmes de logistique d'urgence, où une réponse rapide au besoin urgent de secours est nécessaire dans les zones touchées immédiatement après une catastrophe (comme les tremblements de terre, les tsunamis, les glissements de terrain, entre autres). En raison des conséquences de ces catastrophes, il est difficile d'estimer avec précision la demande de matériel de secours ou les temps de déplacement entre les centres de secours et les zones sinistrées (Sheu (2007)).

Deux approches peuvent être envisagées selon que les allocations des nœuds de demande aux centres sont effectuées avant (voir par exemple Averbakh and Berman (1997), Lu (2013)) ou après (voir par exemple Du et al. (2020); Demange et al. (2020)) la révélation de l'incertitude. Le premier cas correspond à des problèmes à une étape, tandis que le second conduit à des problèmes à deux étapes dans lesquels les allocations des nœuds de demande sont des variables de recours. Dans ce contexte, la plupart des travaux se sont concentrés sur l'étude de la modélisation de la programmation en nombres entiers et des approches heuristiques de résolution (voir par exemple, Baron et al. (2011); Hasani and Mokhtari (2018); Paul and Wang (2015); Trivedi and Singh (2017, 2019)).

A notre connaissance, ce travail est le premier à étudier la solution exacte du problème robuste en deux étapes du p-centre pondéré ($RPCP_2$), avec $p > 1$, dans lequel l'incertitude sur les distance et les demandes des nœuds est modélisée par des ensembles d'incertitude par intervales.

Contributions et conclusions

Nous étudions la solution d'un problème robuste du p-centre, en tenant compte de l'incertitude dans les demandes de nœuds et les longueurs d'arêtes avec des ensembles d'intervalles d'incertitude. Deux variantes de ce problème sont possibles selon que les allocations des nœuds de demande aux centres sont effectuées après que l'incertitude a été révélée ($RPCP_2$) ou non ($RPCP_1$).

De même que pour ($RPCP_1$), il est prouvé que pour ($RPCP_2$) un sous-ensemble fini de scénarios de l'ensemble d'incertitude de la boîte peut être pris en compte sans perdre l'optimalité. Ce résultat est utilisé pour proposer cinq reformulations robustes basées sur différentes formulations MILP du problème du p-centre déterministe. Pour résoudre ces reformulations de manière optimale, un algorithme de *génération de colonnes et de contraintes* et un algorithme de *branch-and-cut* sont introduits. De plus, on identifie une borne inférieure sur la valeur optimale du problème déterministe du centre p associé au sous-ensemble fini de scénarios considérés. Ce résultat est utilisé pour réduire de manière significative le temps de résolution des algorithmes proposés. Enfin, il est souligné comment les méthodes proposées peuvent être adaptées pour résoudre de manière optimale ($RPCP_1$).

Une étude numérique est présentée pour comparer les performances des algorithmes sur une étude de cas, sur des instances générées aléatoirement, et sur quelques instances de l'ORLIB. Il a été possible de résoudre de manière optimale les 68 instances considérées. L'algorithme *génération de colonnes et de contraintes* basé sur les formulations ($RF3$) et ($RF5$) est plus efficace que celui basé sur ($RF1$), ($RF2$) et ($RF4$). En effet, l'ajout d'un scénario ne nécessite pas l'ajout d'une variable. Cette formulation permet la mise en œuvre d'un algorithme de type *branch-and-cut* qui réduit considérablement le temps de résolution.

Dans les travaux futurs, l'analyse d'instances plus importantes avec d'autres ensembles d'incertitude de boîtes aléatoires pourrait être envisagée. Pour améliorer encore les performances de l'algorithme *branch-and-cut*, d'autres stratégies de branchement pourraient être évaluées et des coupes d'intégralité (UserCuts) pourraient être générées dynamiquement. Les algorithmes pourraient également être améliorés en résolvant le (PCP) déterministe à chaque itération avec une autre méthode exacte telle que celle de Contardo et al. (2019) ou Gaar and Sinnl (2022).

Contents

List of Figures

List of Tables

Chapter 1

Introduction

One of the most important branches of *operations research* and the main topic of this thesis is the study of *location problems*, specifically *discrete* location problems. These problems are present in a wide variety of applications, such as clustering, districting, routing, and the location of facilities such as warehouses, industrial plants, gas stations, police stations, or power plants. The study of these problems contributes to improve resource allocation, reduce costs and improve service provisions in various communities and industries. Moreover, including uncertainty in the modeling of these problems allows to anticipate decision making to face various adverse scenarios. That is the case in the context of natural disasters like earthquakes or hurricanes for which these problems can assist identifying optimal locations for temporary shelters, medical centers, and distribution points for essential supplies.

Solving discrete location problems can be challenging due to their size and computational complexity inherent to their exponential combinatorial structure, most of them are NP-Hard problems. Although there are several generic solvers such as CPLEX or Gurobi that can solve many of these problems, the computational complexity quickly becomes intractable as the instance size increases. In this context, several solution methods have been developed to deal with these limitations, but most of them are approximation methods. Thus, proposing new or improving existing exact solution methods for large-scale instances is constantly a challenge.

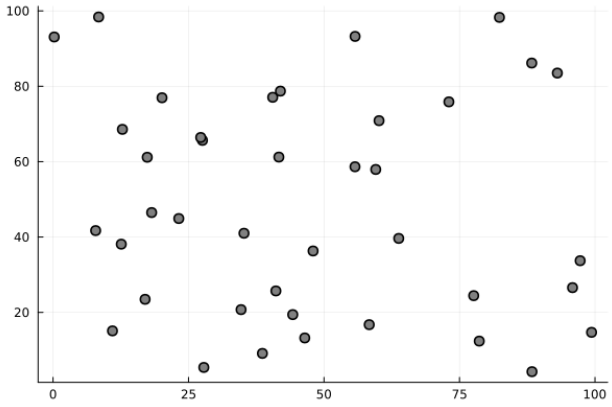
In this chapter, we present the discrete location problems studied in this research, the main contributions, and the outline of this thesis.

1.1 The discrete p-facility location problems

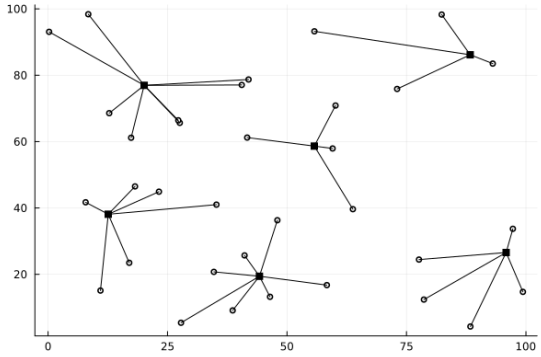
There are four elements that describe every location problem: (1) clients, that presumably are previously located, (2) facilities to be located, (3) a space where facilities and clients are located, and (4) a metric that indicates cost, time or distance between facilities and clients (ReVelle and Eiselt (2005)). Among the fundamental discrete location problems are the *Facility Location Problem (FLP)*, the *Uncapacitated Facility Location Problem (UFLP)*, and the *Capacitated Facility Location Problem (CFLP)* (Laporte et al. (2019b)). (FLP) involves selecting the optimal locations of a predefined number of facilities to satisfy the demand of the clients. The objective is to minimize the sum of transportation and set-up costs. $(CFLP)$ is a variant of (FLP) in which facilities have limited capacities to satisfy clients. Finally, in the $(UFLP)$, neither facility capacities nor client demands are considered.

This research studies "*p-facility problems*" in which there is a fixed number of p facilities to be located usually without taking into account opening or operating costs. We exploit this particular structure for the exact solution of large-scale instances of the *p-median problem (PMP)* and the *p-center problem (PCP)*. These two problems only differ in their objective function. The (PMP) consists in locating p facilities such that the sum of the allocation distances of all demand points to their closest facility is minimized. Instead, the (PCP) seeks to minimize the largest allocation distance.

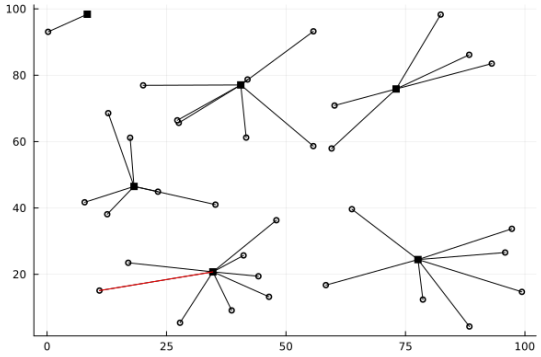
Figure 1.1 illustrates optimal solutions of both problems for a single instance with 40 clients and candidate sites, and p equal to 6. This instance is said to be symmetric since the candidate sites and the clients locations are identical. Note that in the (PCP) solution (Figure 1.1c) a facility is allocated to only 2 isolated clients to reduce the maximum allocation distance (the red line) of 33 to 24. In contrast, the sum of the allocation distances increases from 530 to 586. The (PCP) is said to be more equitable as it will reduce the disparity in allocation distances. On the other hand, (PMP) is said to be more efficient since the average allocation distance is minimized.



(a) Symmetric instance with 40 nodes from a 100×100 square



(b) p-median solution



(c) p-center solution

Figure 1.1: Comparison of optimal (PMP) and (PCP) solutions for same instance (a) with $p = 6$

1.2 Contributions and structure of the manuscript

In this thesis, we propose mathematical models and implement different exact solution methods for three p-facility problems. We have devoted a chapter to each of these problems, which are organized as follows.

Firstly in Chapter 2, we present theoretical framework of this research with the most important definitions and methodologies used throughout the other chapters. We provide a brief description of the most relevant concepts of Mixed-Integer Linear Programming (MILP), MILP solution methods, and optimization under uncertainty.

In Chapter 3, we study the *p-median* problem. We focus on the most efficient MILP formulation. We develop an algorithm based on the Benders decomposition that outperforms the state-of-the-art exact methods. Our method considers a two-stage approach and an efficient algorithm for the separation of the Benders cuts. The computational results are presented on more than 230 benchmark instances with up to 238025 clients and sites. Many instances are solved optimally for the first time or their best known values are improved. We show that our

implementation can also easily be applied to the (UFL). This work has been published in the journal *European Journal of Operational Research* (Duran-Mateluna et al. (2023a)).

The *p-center* problem is considered in Chapter 4. We first compare the five main MILP formulations from the literature. We study two Benders decompositions on one of the most efficient formulations. We show that a new Benders reformulations with more general cuts can be derived from this decomposition. We also propose an alternative exact algorithm based on a clustering procedure using a feature relying on the structure of the problem. All proposed and state-of-the-art methods are compared in benchmark instances. The obtained results are analyzed, highlighting the advantages and disadvantages of each method.

In Chapter 5, we study a *two-stage robust p-center* problem with uncertainty on the node demands and distances. We introduce the robust reformulation of the p-center problem based on the five main deterministic MILP formulations introduced in Chapter 4. We prove that only a finite subset of scenarios from the infinite box uncertainty set can be considered without losing optimality. We also propose a *column-and-constraint generation* algorithm and a *branch-and-cut* algorithm to efficiently solve this problem. We highlight how these algorithms can also be adapted to solve the *robust single-stage* problem. The different proposed formulations are tested on randomly generated instances and on a case study from the literature. This work has been published in the special issue on *Recent Advances in Location Science* of the journal *Computers & Operations Research* (Duran-Mateluna et al. (2023b)).

Finally, Chapter 6 provides a general conclusion and discusses some directions for further research.

Chapter 2

Theoretical background

The main theoretical framework of this research is *Combinatorial Optimization*. This chapter presents the necessary background on the solution techniques and approaches used throughout this thesis.

2.1 MILP problems

Our research focuses on the study of different formulations of discrete location problems using Mixed-Integer Linear Programming (MILP problems). The following are some basic definitions.

- **Polyhedron**: A polyhedron P is a set of points in $\mathbb{R}^n$ satisfying a finite number of linear inequalities, i.e., $P = \{x \in \mathbb{R}^n : Ax \geq b\}$.
- **Polytope**: A polytope is a bounded polyhedron.
- **Valid Inequality**: Given a polyhedron P in $\mathbb{R}^n$. An inequality $a^T x \geq \alpha$ is valid for P if it is verified by every point of P, i.e., $P \subseteq \{x \in \mathbb{R}^n : a^T x \geq \alpha\}$.
- **Convex combination**: Given a set of points $S = \{x_1, ...x_k\}$ in $\mathbb{R}^n$. A point $x \in \mathbb{R}^n$ can be obtained by convex combination of points of S if there exist positive scalars $\lambda_1, ..., \lambda_k$ such that

$$x = \sum_{i=1}^{k} \lambda_i x_i$$
$$\sum_{i=1}^{k} \lambda_i = 1$$
$$\lambda_i \geq 0 \qquad i = 1, ..., k$$

- **Convex set**: A set of points in $\mathbb{R}^n$ is convex if it contains all convex combinations of its points.

- **Convex hull**: Given a set of points $S = \{x_1, ...x_k\}$ in $\mathbb{R}^n$. The convex hull of the points of S noted by $conv(S)$ is the smallest convex set that includes S.
- **Affine independence**: Given a set of points $S = \{x_1, ...x_k\}$ in $\mathbb{R}^n$. These k points are said to be affinely independent if the following system

$$\sum_{i=1}^{k} \lambda_i x_i = 0$$
$$\sum_{i=1}^{k} \lambda_i = 0$$

  has as its unique solution $\lambda_i = 0$ for $i = 1, ..., k$.
- **Dimension**: A polyhedron P in $\mathbb{R}^n$ is of dimension k, denoted $dim(P) = k$, if the maximum number of affinely independent points of P is $k + 1$.
- **Face**: Given a polyhedron $P = \{x \in \mathbb{R}^n : Ax \geq B\}$ in $\mathbb{R}^n$ and $a^T x \geq \alpha$ a valid inequality for P. The set $F = \{x \in P : a^T x = \alpha\}$ is said to be a face of P.
- **Facet**: A face F of a polyhedron P is a facet of P if $dim(F) = dim(P) - 1$.
- **Extreme Point**: Given a polyhedron P. A point $x \in P$ is an extreme point of P if it cannot be obtained as a convex combination of other points in P, x is a facet of P of dimension 0.
- **Separation problem**: The separation problem associated with a linear system $Ax \leq b$ and a vector y consists in verifying whether y is a solution of $Ax \leq b$ and if not in finding a constraint of this system violated by y.

2.2 MILP solution methods

The following are the fundamental methods for solving MILP problems, which are used in the literature and that we also consider.

- **Branch-and-Bound** ($B\&B$): Partition the feasible solution space into smaller sub-spaces (branches in a search tree) and solve the linear relaxation at each node. Integer feasible solutions are progressively found as the search tree is narrowed down. ($B\&B$) incorporates pruning strategies to avoid exploring unnecessary branches.
- **Cutting Plane Methods**: Iteratively refine the solution space with linear inequalities, called cuts.

- **Branch-and-Cut** ($B\&C$): An enhancement of the ($B\&B$) approach that incorporates cutting planes to strengthen the linear relaxations at each node.
- **Column generation**: First solve the problem with only a subset of its variables. Then iteratively, variables that have the potential to improve the objective function are added to the program until no further improvement is possible.
- **Branch-and-Price** ($B\&P$): Embeds a column generation in the branch-and-bound search tree. At each node the fractional solution is evaluated to determine whether a new variable must be added to the problem.

2.2.1 Benders decomposition method

Another classical exact method is the *Benders Decomposition*. Introduced by Benders (1962), it is particularly valuable for tackling complex problems involving a mix of integer and continuous variables. We explore the Benders decomposition in Chapters 3 and 4 for the p-median and p-center problems, respectively.

The core idea of Benders decomposition is to split the problem into a *master problem* and one or several *sub-problems*. The master problem is a relaxation of the original problem, where certain constraints are omitted, making it easier to solve. The sub-problem tries to set the value of the omitted variables according to the solution of the master problem. If this solution is not optimal, they generate cuts and the master problem is solved again. Several improvements have been developed to accelerate the efficiency of this classical algorithm. A detailed literature review is presented in Rahmaniani et al. (2017).

The Benders decomposition showed good results on discrete location problems. It was already studied on the *Uncapacitated Facility Location Problem* (UFL) in Cornuejols et al. (1980) and Magnanti and Wong (1981). Most recently, Fischetti et al. (2017) propose a Benders decomposition method within a *branch-and-cut* approach to efficiently solve very large size instances of the (UFL). Cordeau et al. (2019) described Benders decomposition for two problems: the *Maximal Covering Location Problem* ($MCLP$), which requires finding a subset of facilities that maximizes the amount of client demands covered while respecting a budget constraint on the cost of the facilities; and the *Partial Set Covering Location Problem* ($PSCLP$), which minimizes the cost of the opened facilities while forcing a certain amount of client demand to be covered. They study a decomposition approach of the two problems based on a *branch-and-Benders-cut* reformulation. Their approach is more efficient when the number of clients is much larger than the number of potential facility locations. Gaar and Sinnl (2022)

also perform a Benders decomposition on the *p-center problem* (PCP) combined with other theoretical results that allow to solve large instances which up to 744,710 nodes. This approach is highly efficient for small values of p.

The following is a brief formal presentation of the method. Let us consider a MILP problem (P) with n integer variables and m continuous variables of the following generic form.

$$(P)\begin{cases} \min & f^T y + c^T x \\ \text{s.t.} & By \geq b \\ & Wy + Tx \geq h \\ & y \in \mathbb{Z}^n_+ \\ & x \in \mathbb{R}^m_+ \end{cases}$$

where $f \in \mathbb{R}^n$, $B \in \mathbb{R}^{k \times n}$, $b \in \mathbb{R}^k$, $c \in \mathbb{R}^m$, $W \in \mathbb{R}^{l \times n}$, $h \in \mathbb{R}^l$, $T \in \mathbb{R}^{l \times m}$. To solve (P) with the Benders decomposition method, it is necessary to introduce an auxiliary variable θ to formulate the following master problem (MP).

$$(MP)\begin{cases} \min\limits_{(y,\theta)} & f^T y + \theta \\ \text{s.t.} & By \geq b \\ & \theta \geq \bar{\theta} \\ & y \in \mathbb{Z}^n_+ \\ & \theta \in \mathbb{R} \end{cases}$$

where $\bar{\theta}$ is a lower bound on θ to ensure that the problem is bounded. The problem (MP) is less restricted than (P), henceforth its objective value defines lower bounds for (P).

Let $\bar{y}$ be a solution of (MP). We can define the following sub-problem (SP):

$$(SP)\begin{cases} \min\limits_{x} & c^T x \\ \text{s.t.} & Wx \geq h - T\bar{y} \\ & x \in \mathbb{R}^m_+ \end{cases}$$

which its dual problem is:

$$(DSP)\begin{cases} \max\limits_{v} & (h - T\bar{y})^T v \\ \text{s.t.} & W^T v \leq c \\ & v \in \mathbb{R}^l_+ \end{cases}$$

If the sub-problem is feasible for $\bar{y}$, its dual sub-problem is also feasible. Thus, the solution $\bar{y}$ is a feasible solution to (P) and we can deduce an upper bound of the original problem (P). Moreover, since the feasible region of the dual sub-problem does not depend on $\bar{y}$. For any extreme points v^* of the polyhedron of (DSP), the following *optimality cut* is: valid for the (MP).

$$\theta \geq (h - Ty)^T v^* \tag{2.1}$$

If the sub-problem is infeasible, its associated dual problem is unbounded. Therefore, there exists an extreme ray r^* of the polyhedron of (DSP), along which the objective function can always be improved. In this case, the following *feasibility cut* can be added to the (MP).

$$0 \geq (h - Ty)^T r^* \tag{2.2}$$

Using the above optimality and feasibility cuts, the original problem (P) can be reformulated as:

$$(RP) \begin{cases} \min & f^T y + \theta & \\ \text{s.t.} & By \geq b & \\ & \theta \geq (h - Ty)^T v^* & \forall\, v^* \in\ \mathcal{P} \subseteq U \\ & 0 \geq (h - Ty)^T r^* & \forall\, r^* \in\ \mathcal{R} \subseteq U \\ & y \in Y & \end{cases}$$

where $\mathcal{P}$ and $\mathcal{R}$ are respectively, the subsets of the extreme points and the extreme rays of polyhedron U of (DSP).

Generally there is an exponential number of extreme points and rays. So a direct solution of the (RP) is impractical. The classic Benders decomposition algorithm to solve (P) is presented in Algorithm 1. A solution of the master problem gives us a lower bound (LB), while a solution of the sub-problem gives us an upper bound (UB) on the optimal value of (P). Once the upper bound and the lower bound are close enough, the algorithm ends.

Algorithm 1: Clasic Benders decomposition algorithm

input: A MILP problem (P)

$(LB, UB) \leftarrow (-\infty, \infty)$

while $UB - LB > \epsilon$ **do**

  $\bar{y} \leftarrow$ Solve (MP)

  if *(MP) is infeasible* **then**

    return Infeasible

  $LB \leftarrow \min(LB,$ (MP) solution value$)$

  Solve (DSP) with the (MP) solution $\bar{y}$

  if *(DSP) is unbounded* **then**

    Find a ray r^*

    Add a feasibility cut with r^* to the (MP)

  else

    Find an extreme point v^* (the solution of the dual sub-problem)

    Add an optimality cut with v^* to the (MP)

    $UB \leftarrow \min(UB,$ (SP) solution value$)$

return $\hat{x}$ and $\bar{y}$

2.3 Optimization under uncertainty

The study of location problems under uncertainty is an important area of research (see e.g. An et al. (2014), Laporte et al. (2019a), and Cheng et al. (2021)). In Chapter 5, we study the p-center problem under uncertainty in two sets of parameters.

We refer to the work Bertsimas and Sim (2004), and Ben-Tal et al. (2009) for more details on the approaches, methods and algorithms for optimization under uncertainty. The following is a brief description of the most important modeling approaches. Let us consider the following deterministic linear problem.

$$\begin{aligned} \min \quad & c^T x \\ \text{s.t} \quad & Ax \leq b \\ & x \in \mathbb{R}^n \end{aligned}$$

where (A, b, c) are the parameters of the problem.

The first aspect to consider in modeling uncertainty is whether the probability distribution of the uncertain parameters is available or can be estimated. If this is the case, the problem can be formulated through *Stochastic Optimization* by optimizing an expected objective value.

$$\begin{aligned} \min \quad & \mathbb{E}_{w \in \mathcal{W}}[Q(x,w)] \\ \text{s.t} \quad & Ax \leq b \\ & x \in \mathbb{R}^n \end{aligned}$$

where $Q(x,w)$ corresponds to the cost of solution x under *scenario* w, which is the realization of the uncertain parameters from a given uncertainty set W. Stochastic programming allows decision-makers to explicitly consider the probabilities of various scenarios and make decisions that balance potential risks and rewards.

Another approach when the probability distribution is known is named the *Chance-Constrained Programming*. In this approach, optimization is performed subject to constraints that ensure a certain probability of meeting specified criteria.

$$\begin{aligned} \min \quad & c^T x \\ \text{s.t} \quad & \mathbb{P}\{a_i(w)x \leq b_i(w) \ \forall i = 1, ..., m\} \geq \alpha \\ & x \in \mathbb{R}^n \end{aligned}$$

here m is the number of linear constraints, and α is the desired probability of occurrence. It is appropriate when the decision-makers want to avoid undesirable events or impose that a desirable event occurs with high probability, but it is usually difficult to solve.

When there is a set of possible probability distributions for the uncertain parameters, it is possible to consider the *Distributionally Robust Optimization* using the following formulation.

$$\begin{aligned} \min \quad \max_{\mathbb{P} \in \mathcal{P}} \quad & \mathbb{E}^{\mathbb{P}}_{w \in \mathcal{W}}[Q(x,w)] \\ \text{s.t} \quad & Ax \leq b \\ & x \in \mathbb{R}^n \end{aligned}$$

where $\mathcal{P}$ is a family of probability distributions. This approach seeks solutions that perform well across all distributions within $\mathcal{P}$.

Finally, when information on the probability distribution is not available or the decision makers are risk-averse, we can consider the *Robust Optimization*. Here, it is only known that each uncertain parameter can belong to an uncertainty set that can be modeled in different ways such as boxes, polyhedral, ellipsoidal, among others. In the case of individual uncertainty sets, the following model can be considered.

$$\begin{aligned} \min \quad & \max_{c \in U_c} \quad & c^T x & \\ \text{s.t} \quad & & a_i(w)x \leq b_i(w) \qquad & \forall\, a_i \in U_{a_i}, \forall\, b_i \in U_{b_i}, \forall\, i = 1, ..., m \\ & & x \in \mathbb{R}^n & \end{aligned}$$

where U_{a_i}, U_{b_i}, U_c are given uncertainty sets. The approach aims to minimize the cost of the *worst-case* scenario. Therefore, it is highly relevant in situations where the lives of people are at stake, for example.

An important aspect to the formulation of these optimization approaches is the moment at which decisions are taken concerning the occurrence of uncertainty. There exists three main approaches. In the *single-stage* optimization approach all decisions are taken before the uncertainty is revealed ("here-and-now" decisions). In the *two-stage* approach decisions are made before and after the uncertainty is revealed. The decision variables of the second stage are called "wait-and-see" decisions, associated to *recourse* variables. Lastly, the *multi-stage* approach is the generalization of the two-stage approach where the decision variables can be taken in more than two phases. This framework can address problems in which the uncertainty is progressively revealed in more than one step. The choice of the approach to be used depends on factors such as the complexity of the problem, the availability of information about the uncertainty, the risk attitude of decision-makers, and the desired trade-off between optimality and robustness.

Chapter 3

Efficient Benders decomposition method for the p-median problem

3.1 Introduction

The *p-median problem* (PMP) is one of the fundamental problems in Location Science (Laporte et al. (2019b)). In the (PMP), the p sites have to be chosen from the set of candidate sites without considering set-up costs. The allocation costs are usually equal to the distance or travel time between clients and sites. Figure 3.1 illustrates an optimal solution of (PMP) for different values of p considering the symmetric instance presented in Figure 1.1a. This shows how different clusters are being identified. As p increases, each cluster has fewer nodes, and thus the average allocation distance decreases.

More formally, we consider a set of N clients, a set of M potential centers to open, and their corresponding index set $\mathcal{N} = \{1, ..., N\}$, and $\mathcal{M} = \{1, ..., M\}$. Let d_{ij} be the distance between client $i \in \mathcal{N}$ and site $j \in \mathcal{M}$, and let $p \in \mathbb{N}$ be the number of sites to open. The objective is to find a set S of p sites such that the sum of the distances between each client and its closest site in S is minimized. The (PMP) is an NP-hard problem (Kariv and Hakimi (1979)) and leads to applications where the sites correspond to warehouses, plants, shelters, etc. This problem occurs in the contexts of emergency logistics and humanitarian relief (An et al. (2014); Mu and Tong (2020); Takedomi et al. (2022)). A well-known clustering problem called *k-medoids problem* is also a special case of the (PMP) in which the set of clients and sites are identical. In this problem, sub-groups of objects, variables, persons, etc. are identified according to defined criteria of proximity or similarity (Klastorin (1985); Park and Jun (2009); Marín and Pelegrín (2019); Ushakov and Vasilyev (2021); Voevodski (2021)).

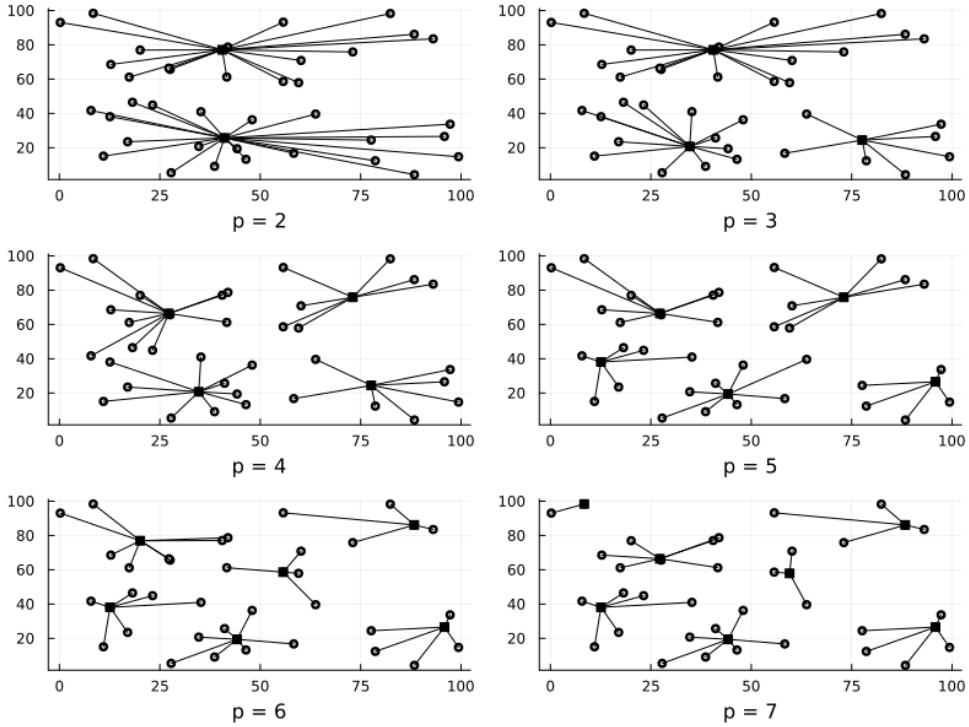


Figure 3.1: (PMP) solutions with different p values for a symmetric instance of 40 nodes.

A great interest in solving large location problems has led to the development of various heuristics and meta-heuristics in the literature. However, the exact solution of large instances remains a challenge. As detailed in Section 2.2.1, some location problems have recently been efficiently solved using the Benders decomposition method within a *branch-and-cut* approach (see e.g., Fischetti et al. (2017); Cordeau et al. (2019); Gaar and Sinnl (2022)). Among them, the *Uncapacitated Facility Location Problem* (UFL). In the (UFL) the number of sites to be opened is not fixed, but an opening cost is associated with each site.

Contribution

This chapter explores a Benders decomposition for the (PMP). We propose an efficient algorithm for the separation of its Benders cuts. We implement a two-phase Benders decomposition algorithm which provides better results than the best exact solution method in the literature `Zebra` (García et al. (2011)). We present the computational results on about 230 benchmark (PMP) instances of different sizes (up to 238,025 clients and sites) satisfying or not the triangle inequality. Finally, our solution method is extended to solve the (UFL).

The rest of the chapter is organized as follows: Section 3.2 presents the literature review of the (PMP). Section 3.3 describes our Benders decomposition method. Section 3.4 presents the computational results. In Section 3.5 some conclusions and research perspectives are draw.

3.2 Literature review

The (PMP) was introduced by Hakimi (1964) where the problem was defined on a graph such that a client can only be allocated to an open neighbor site. Since then, exact and approximation methods have been developed to solve the problem, as well as a wide variety of variants and extensions. We refer to Marín and Pelegrín (2019) for more details and references. The following is a summary of the main formulations of this problem and its state-of-the-art exact solution methods.

3.2.1 MILP formulations

The classical mathematical programming formulation for the (PMP) was proposed by ReVelle and Swain (1970) who formulated the problem with a binary variable y_j for each site $j \in \mathcal{M}$ that takes value of 1, if the site is open, and 0 otherwise; and a binary variable x_{ij} that takes value of 1 if client $i \in \mathcal{N}$ is allocated to site $j \in \mathcal{M}$ and 0 otherwise.

$$
(F1)\begin{cases}
\min & \displaystyle\sum_{i\in\mathcal{N}}\sum_{j\in\mathcal{M}} d_{ij}x_{ij} & & (3.1)\\
\text{s.t.} & \displaystyle\sum_{j\in\mathcal{M}} y_j = p & & (3.2)\\
& \displaystyle\sum_{j\in\mathcal{M}} x_{ij} = 1 & i\in\mathcal{N} & (3.3)\\
& x_{ij} \le y_j & i\in\mathcal{N},\ j\in\mathcal{M} & (3.4)\\
& x_{ij} \ge 0 & i\in\mathcal{N},\ j\in\mathcal{M} & (3.5)\\
& y_j \in \{0,1\} & j\in\mathcal{M} &
\end{cases}
$$

Constraint (3.2) fixes the number of open sites to p. Constraints (3.3) ensure that each client is allocated to exactly one site, and Constraints (3.4) ensure that no client is allocated to a closed site. The binary variables x_{ij} can actually be relaxed as in Constraints (3.5).

An alternative formulation $(F2)$ was proposed by Cornuejols et al. (1980) which orders for each client all its distinct distances to the sites. More formally, for any client $i \in \mathcal{N}$, let $K_i \le M$ be the number of different distances from i to any site. Let $D_i^1 < D_i^2 < ... < D_i^{Ki}$ be these distances sorted, and $\mathcal{K}_i$ the corresponding index set. Formulation $(F2)$ uses the same y variables as in formulation $(F1)$ and introduces new binary variables z. For any client $i \in \mathcal{N}$ and $k \in \mathcal{K}_i$, $z_i^k = 0$ if and only if there is an open site at distance at most D_i^k from client i.

$$
(F2)\begin{cases}
\min & \displaystyle\sum_{i\in\mathcal{N}}\Big(D_i^1+\sum_{k=1}^{K_i-1}(D_i^{k+1}-D_i^k)z_i^k\Big) & & (3.6)\\
\text{s.t.} & \displaystyle\sum_{j\in\mathcal{M}} y_j = p & & (3.7)\\
& \displaystyle z_i^k+\sum_{j:d_{ij}\le D_i^k} y_j \ge 1 & i\in\mathcal{N},\ k\in\mathcal{K}_i & (3.8)\\
& z_i^k \ge 0 & i\in\mathcal{N},\ k\in\mathcal{K}_i & (3.9)\\
& y_j\in\{0,1\} & j\in\mathcal{M} &
\end{cases}
$$

Objective (3.6) minimizes the sum of the allocation distances over all clients. Constraints (3.8) ensure that variable z_i^k takes the value 1 if there is no site at a distance less than or equal to D_i^k of client $i\in\mathcal{N}$. In that case $(D_i^{k+1}-D_i^k)$ is added to the objective. Otherwise, given the positive coefficients in the objective function, z_i^k takes the value 0. Here again, the binary variables z_i^k can be relaxed as in Constraints (3.9).

Formulation $(F2)$ can be much smaller than $(F1)$ and both have the same linear relaxation value (Cornuejols et al. (1980)). Formulation $(F1)$ contains $N\times M$ variables x and $1+N+N\times M$ constraints while $(F2)$ contains $K=\sum_{i\in\mathcal{N}}K_i$ variables z and $K+1$ constraints. As $K\le N\times M$, it follows that $(F2)$ has at most as many variables and constraints as $(F1)$. Usually K is significantly smaller than $N\times M$.

Elloumi (2010) introduced another formulation based on $(F2)$. Given that, by definition, z_i^{k-1} equal to 0 implies that z_i^k is also equal to 0, Constraints (3.8) can be replaced by (3.12) and (3.13).

$$
(F3)\begin{cases}
\min & \displaystyle\sum_{i\in\mathcal{N}}\Big(D_i^1+\sum_{k=1}^{K_i-1}(D_i^{k+1}-D_i^k)z_i^k\Big) & & (3.10)\\
\text{s.t.} & \displaystyle\sum_{j\in\mathcal{M}} y_j = p & & (3.11)\\
& \displaystyle z_i^1+\sum_{j:d_{ij}=D_i^1} y_j \ge 1 & i\in\mathcal{N} & (3.12)\\
& \displaystyle z_i^k+\sum_{j:d_{ij}=D_i^k} y_j \ge z_i^{k-1} & i\in\mathcal{N},\ k=2,...,K_i & (3.13)\\
& z_i^k \ge 0 & i\in\mathcal{N},\ k\in\mathcal{K}_i & (3.14)\\
& y_j\in\{0,1\} & j\in\mathcal{M} & (3.15)
\end{cases}
$$

Constraints (3.12) correspond to Constraints (3.8) for $k=1$. Constraints (3.13) ensure that z_i^k takes the value 1, if $z_i^{k-1}=1$ and there is no open site at distance D_i^k exactly from i. Formulations $(F2)$ and $(F3)$ use the same set of variables y and z, have exactly the same objective function, and have the same linear relaxation bound (Elloumi (2010)). However, $(F3)$ has much more

zeros in the constraint coefficient matrix, which makes it perform significantly better than $(F2)$. Therefore, $(F3)$ is considered for our Benders decomposition.

3.2.2 Solution methods

The literature contains many solution methods for the (PMP). The main heuristics are presented in the following surveys: Reese (2006); Mladenović et al. (2007); Basu et al. (2015); Irawan and Salhi (2015a). In the following, only the most relevant exact solution methods are mentioned.

Galvão (1980) solved the (PMP) within a *branch-and-bound* framework solving many linear relaxations of sub-problems of size $N = 30$ using formulation $(F1)$. He then devised a method to efficiently obtain good lower bounds instead of optimally solving the relaxed continuous sub-problems.

Avella et al. (2007) designed a *branch-and-cut-and-price* algorithm also based on $(F1)$ that was able to solve instances with up to $N = 5535$. Cuts were added based on valid inequalities called lifted odd hole and cycle inequalities. Pricing was carried out by solving a master problem to optimality and using dual variables to price out the variables of the initial problem that were not considered in the master, adding new variables if necessary. In this approach, Constraints (3.3) were also relaxed and incorporated to the master problem when the corresponding column was.

García et al. (2011) considered a cut and column generation algorithm based on formulation $(F2)$. The main idea, also presented in Elloumi (2010) on formulation $(F3)$ and implemented in Elloumi and Plateau (2010), relies on the property that the z variables satisfy $z_i^k \geq z_i^{k+1}$ in any optimal solution of $(F2)$ or its LP relaxation. Therefore, it is enough to solve these problems on a reduced subset of variables z, keep enlarging this subset, and stop as soon as one is sure that the remaining z variables can be set to zero to get an optimal solution. This idea is implemented within a *branch-and-cut-and-price* method that the authors name `Zebra`. It starts with a very small set of z variables and constraints, and adds more when necessary. `Zebra` is an exact solution method that performed well on instances with up to $N = 85,900$ with large values of p.

3.3 Benders decomposition for the *(PMP)*

In this section, we present a Benders decomposition for the *(PMP)* based on formulation $(F3)$. More details and references to this method are presented in the Section 2.2.1. We show that there is a finite number of Benders cuts and that they can be separated using an efficient algorithm.

3.3.1 Formulation

For a fixed solution y, the problem decomposes into N sub-problems. Each one computes the allocation distance of a client. In the master problem, the z variables are removed and a new set of continuous variables θ_i representing the allocation distance of each client $i \in \mathcal{N}$ is introduced:

$$(MP) \begin{cases} \min & \displaystyle\sum_{i \in \mathcal{N}} \theta_i & \\ \text{s.t.} & \displaystyle\sum_{j \in \mathcal{M}} y_j = p & \\ & \theta_i \text{ satisfies } BD_i & i \in \mathcal{N} \\ & y_j \in \{0,1\} & j \in \mathcal{M} \end{cases}$$

where BD_i is the set of Benders cuts associated to client $i \in \mathcal{N}$. This set is initially empty and grows through the iterations.

The sub-problem for each client $i \in \mathcal{N}$ associated to a feasible solution $\bar{y}$ of (MP) is defined by:

$$(SP_i(\bar{y})) \begin{cases} \min & \displaystyle D_i^1 + \sum_{k=1}^{K_i - 1} (D_i^{k+1} - D_i^k) z_i^k & \\ \text{s.t.} & \displaystyle z_i^1 \geq 1 - \sum_{j : d_{ij} = D_i^1} \bar{y}_j & \\ & \displaystyle z_i^k - z_i^{k-1} \geq - \sum_{j : d_{ij} = D_i^k} \bar{y}_j & k \in \{2, ..., K_i\} \\ & z_i^k \geq 0 & k \in \mathcal{K}_i \end{cases}$$

and its corresponding dual sub-problem is:

$$(DSP_i(\bar{y})) \begin{cases} \max & \displaystyle D_i^1 + v_i^1 (1 - \sum_{j : d_{ij} = D_i^1} \bar{y}_j) - \sum_{k=2}^{K_i} v_i^k \sum_{j : d_{ij} = D_i^k} \bar{y}_j & \\ \text{s.t.} & v_i^k - v_i^{k+1} \leq D_i^{k+1} - D_i^k & k \in \{1, ..., K_i - 1\} \\ & v_i^k \geq 0 & k \in \mathcal{K}_i \end{cases}$$

Note that $(SP_i(\bar{y}))$ and $(DSP_i(\bar{y}))$ are feasible for any $\bar{y}$. From an extreme point $\bar{v}$ of $(DSP_i(\bar{y}))$, the following optimality Benders cut is obtained:

$$\theta_i \geq D_i^1 + \bar{v}_i^1(1 - \sum_{j:d_{ij}=D_i^1} y_j) - \sum_{k=2}^{K_i} \bar{v}_i^k \sum_{j:d_{ij}=D_i^k} y_j \qquad i \in \mathcal{N} \tag{3.16}$$

3.3.2 Separation problem

The performance of Benders decomposition relies on the resolution of the master problem and the sub-problems. In our decomposition, we can have a large number of sub-problems to solve since it is equal to the number of clients at each iteration. Below, we show that the sub-problems can be solved efficiently.

Lemma 3.3.1. *Given a solution $\bar{y}$ of the master problem (MP) or of its LP-relaxation, the following solution $\bar{z}_i$ is optimal for $(SP_i(\bar{y}))$:*

$$\bar{z}_i^k = \max\left(0,\ 1 - \sum_{j:d_{ij}\leq D_i^k} \bar{y}_j\right) \quad i \in \mathcal{N},\ k \in \mathcal{K}_i$$

Proof. We can rewrite the constraints of $(SP_i(\bar{y}))$ for each $i \in \mathcal{N}$ as follows:

- $z_i^1 \geq 1 - \sum_{j:d_{ij}=D_i^1} \bar{y}_j$;
- $z_i^2 \geq z_i^1 - \sum_{j:d_{ij}=D_i^2} \bar{y}_j \geq 1 - \sum_{j:d_{ij}=D_i^1} \bar{y}_j - \sum_{j:d_{ij}=D_i^2} \bar{y}_j = 1 - \sum_{j:d_{ij}\leq D_i^2} \bar{y}_j$;
- ...
- $z_i^k \geq z_i^{k-1} - \sum_{j:d_{ij}=D_i^k} \bar{y}_j \geq 1 - \sum_{j:d_{ij}=D_i^{k-1}} \bar{y}_j - \sum_{j:d_{ij}=D_i^k} \bar{y}_j = 1 - \sum_{j:d_{ij}\leq D_i^k} \bar{y}_j$.

Since we minimize an objective function with non-negative coefficients, the $\bar{z}_i^k$ variables are as small as possible in an optimal solution. Thus, setting $\bar{z}_i^k$ to the value $\max\left(0,\ 1 - \sum_{j:d_{ij}\leq D_i^k} \bar{y}_j\right)$ leads to a feasible and optimal solution. □

From Lemma 3.3.1 we observe that the optimal values of variables z_i^k in $(SP_i(\bar{y}))$ are decreasing when k increases. In order to obtain a dual solution, we identify the last strictly positive term of this sequence.

Definition 3.3.1. *Given a solution $\bar{y}$ of the master problem (MP) or of its LP-relaxation, let $\tilde{k}_i$ be the following index:*

$$\tilde{k}_i = \begin{cases} 0 & \text{if } \sum_{j:d_{ij}=D_i^1} \bar{y}_j \geq 1 \\ \max\{k \in \mathcal{K}_i : 1 - \sum_{j:d_{ij}\leq D_i^k} \bar{y}_j > 0\} & \text{otherwise} \end{cases} \qquad i \in \mathcal{N}$$

Note that, if $\bar{y}$ is binary, then the corresponding allocation distance of client $i \in \mathcal{N}$ is $D_i^{\tilde{k}_i+1}$.

Lemma 3.3.2. *Given a solution $\bar{y}$ of the master problem (MP) or of its LP-relaxation and the corresponding indices $\tilde{k}_i$, the optimal value of $SP_i(\bar{y})$ for $i \in \mathcal{N}$ is:*

$$OPT(SP_i(\bar{y})) = \begin{cases} D_i^1 & \text{if } \tilde{k}_i = 0 \\ D_i^{\tilde{k}_i+1} - \sum_{j:d_{ij}\leq D_i^{\tilde{k}_i}} (D_i^{\tilde{k}_i+1} - d_{ij})\bar{y}_j & \text{otherwise} \end{cases} \tag{3.17}$$

Proof. If $\tilde{k}_i = 0$, then using Lemma 3.3.1 and Definition 3.3.1, then all the values $\bar{z}_i^k$ are equal to 0 and $OPT(SP_i(\bar{y})) = 0$. Otherwise, when $\tilde{k}_i \geq 1$, we have the following:

$$\begin{aligned} OPT(SP_i(\bar{y})) &= D_i^1 + \sum_{k=1}^{\tilde{k}_i} (D_i^{k+1} - D_i^k)(1 - \sum_{j:d_{ij}\leq D_i^k} \bar{y}_j) \\ &= D_i^1 + D_i^{\tilde{k}_i+1}(1 - \sum_{j:d_{ij}\leq D_i^{\tilde{k}_i}} \bar{y}_j) - D_i^1(1 - \sum_{j:d_{ij}\leq D_i^1} \bar{y}_j) + \\ &\quad \sum_{k=2}^{\tilde{k}_i} D_i^k((1 - \sum_{j:d_{ij}\leq D_i^{k-1}} \bar{y}_j) - (1 - \sum_{j:d_{ij}\leq D_i^k} \bar{y}_j)) \\ &= D_i^{\tilde{k}_i+1}(1 - \sum_{j:d_{ij}\leq D_i^{\tilde{k}_i}} \bar{y}_j) + \sum_{k=1}^{\tilde{k}_i} \sum_{j:d_{ij}=D_i^k} d_{ij}\bar{y}_j \\ &= D_i^{\tilde{k}_i+1} - \sum_{j:d_{ij}\leq D_i^{\tilde{k}_i}} (D_i^{\tilde{k}_i+1} - d_{ij})\bar{y}_j \end{aligned}$$

□

Proposition 3.3.1. *Given a solution $\bar{y}$ of the master problem (MP) or of its LP-relaxation and the corresponding indices $\tilde{k}_i$, the following solution $\bar{v}_i$ is optimal for $DSP_i(\bar{y})$:*

$$\bar{v}_i^k = \begin{cases} D_i^{\tilde{k}_i+1} - D_i^k, & \text{if } k \leq \tilde{k}_i \\ 0, & \text{otherwise} \end{cases} \qquad i \in \mathcal{N},\ k \in \mathcal{K}_i$$

Proof. From the theorem of complementary slackness, for $i \in \mathcal{N}$ we have:

$$\bar{v}_i^k - \bar{v}_i^{k+1} = D_i^{k+1} - D_i^k \qquad k \leq \tilde{k}_i \tag{3.18}$$

because for $k \leq \tilde{k}_i$, the primal variables values $\bar{z}_i^k$ are strictly positive. Moreover, if we sum Equations (3.18) from $k' = k$ to $k' = \tilde{k}_i$, we obtain

$$\bar{v}_i^k = v_i^{\tilde{k}_i+1} + (D_i^{\tilde{k}_i+1} - D_i^k) \qquad k \leq \tilde{k}_i \tag{3.19}$$

We build a feasible solution of $(DSP_i(\bar{y}))$ by setting $\bar{v}_i^k = 0$ for $k > \tilde{k}_i$ and $\bar{v}_i^k = (D_i^{\tilde{k}_i+1} - D_i^k)$ for $k \leq \tilde{k}_i$. The objective value $\mathcal{V}(\bar{v})$ of this solution is:

$$\begin{aligned}
\mathcal{V}(\bar{v}) &= D_i^1 + (D_i^{\tilde{k}_i+1} - D_i^1)(1 - \sum_{j:d_{ij}=D_i^1} \bar{y}_j) - \sum_{k=2}^{\tilde{k}_i} (D_i^{\tilde{k}_i+1} - D_i^k) \sum_{j:d_{ij}=D_i^k} \bar{y}_j \\
&= D_i^{\tilde{k}_i+1} - \sum_{k=1}^{\tilde{k}_i} (D_i^{\tilde{k}_i+1} - D_i^k) \sum_{j:d_{ij}=D_i^k} \bar{y}_j \\
&= D_i^{\tilde{k}_i+1} - \sum_{k=1}^{\tilde{k}_i} \sum_{j:d_{ij}=D_i^k} (D_i^{k+1} - D_i^k)\bar{y}_j \\
&= D_i^{\tilde{k}_i+1} - \sum_{j:d_{ij} \leq D_i^{\tilde{k}_i}} (D_i^{\tilde{k}_i+1} - d_{ij})\bar{y}_j = OPT(SP_i(\bar{y}))
\end{aligned}$$

Hence, this dual solution is feasible and it has the same objective value as the optimal value of the primal solution. Thus, this solution $\bar{v}$ is optimal for $(DSP_i(\bar{y}))$. □

Corollary 1 The Benders cuts (3.16) can be written as follows:

$$\begin{cases} \theta_i \geq D_i^1 & \text{if } \tilde{k}_i = 0 \\ \theta_i \geq D_i^{\tilde{k}_i+1} - \sum_{j:d_{ij} \leq D_i^{\tilde{k}_i}} (D_i^{\tilde{k}_i+1} - d_{ij}) y_j & \text{otherwise} \end{cases} \tag{3.20}$$

We observe that these inequalities are the same as those obtained in Cornuejols et al. (1980) and Magnanti and Wong (1981) for (PMP) on formulation $(F1)$. This was quite unexpected, since we use the formulation $(F3)$, even though the master problems are the same in the two decompositions, the sub-problems are different. This class of Benders cuts were also presented in Fischetti et al. (2017) for the (UFL) on a similar formulation to $(F1)$.

3.3.3 Efficient separation algorithm

Corollary 3.16 highlights that the Benders cuts can be obtained in polynomial time by computing $\tilde{k}_i$. We use Algorithm 2 to separate Constraints (3.20). For each client $i \in \mathcal{N}$, we first compute $\tilde{k}_i$ and $OPT(SP_i(\bar{y}))$ from the current (MP) solution $(\bar{y},\ \bar{\theta})$ (Steps 3 and 4) thus updating the upper bound UB of (MP) (Step 5). Then, if the value of the allocation distance in the current (MP) solution is underestimated (Step 6), the corresponding Benders cuts (3.20) are directly built (Step 7).

Algorithm 2: Separation algorithm

input :

- Instance data ($\mathcal{N}$, $\mathcal{M}$, $\mathcal{K}_i$, distances $D_i^1, ..., D_i^{K_i}$ and d_{ij} for each $i \in \mathcal{N}$, $j \in \mathcal{M}$)
- Current (MP) solution $(\bar{y},\ \bar{\theta})$

output:

- Upper bound of (MP).

1 $UB \leftarrow 0$
2 **for** $i \in \mathcal{N}$ **do**
3 Compute $\tilde{k}_i$ with Algorithm 3
4 Compute $OPT(SP_i(\bar{y}))$ through Lemma 3.3.2
5 $UB \leftarrow UB + OPT(SP_i(\bar{y}))$
6 **if** $\bar{\theta}_i < OPT(SP_i(\bar{y}))$ **then**
7 Add the corresponding cut (3.20) to (MP)
8 **return** UB

The memory management of the distances between the clients and the sites can be challenging for large-scale instances. For example, in the work of Cornuejols et al. (1980); Magnanti and Wong (1981); Fischetti et al. (2017), the increasingly ordered distances $\{d_{ij}\}_{j\in\mathcal{M}}$ of each client $i \in \mathcal{N}$ are considered as an input.

In our approach, the complexity of Algorithm 2 is determined by the computation of index $\tilde{k}_i$. It can be computed in $O(M)$ by Algorithm 3. This algorithm takes as an input a vector $S_i \in \mathcal{M}^M$ such that S_{ir} is the r^{th} closest site to client $i \in \mathcal{N}$. Hence, $d_{i\mathcal{S}_{ir}}$ is the distance between i and its r^{th} closest site. Afterwards, given index $\tilde{k}_i$, Steps 4 and 5 of Algorithm 2 can be computed in $O(M)$ and $O(1)$, respectively. Then, considering the N clients, a complexity in $O(NM)$ is obtained for Algorithm 2. As in García et al. (2011), the distances $\{d_{ij}\}_{j\in\mathcal{M}}$ of each client $i \in \mathcal{N}$ are calculated as they are needed.

Consequently, the $N \times M$ matrix $\mathcal{S}$ is built only once in a preprocessing step in $O(NMlog(M))$ using the QuickSort algorithm. The computation time of this matrix may be longer than the runtime of the solution method depending on the size of the instances. For example, for instances with 5000, 13,000, 27,000, and 85,000 clients and sites, the computer described below in Section 3.4 builds the matrix on average in 5, 25, 90 and 1100 seconds, respectively. Furthermore, to reduce the memory requirements for storing this matrix $\mathcal{S}$, we consider the fact that a client will never be allocated to one of its furthest p sites. Therefore, the size of the matrix $\mathcal{S}$ is reduced to $N \times (M - p)$.

Algorithm 3: Computing $\tilde{k}_i$

input :

- Instance data ($\mathcal{N}$, $\mathcal{M}$, $\mathcal{K}_i$, $\mathcal{S}$ matrix, and distances d_{ij} for $i \in \mathcal{N}$, $j \in \mathcal{M}$)
- Current (MP) solution $\bar{y}$
- $i \in \mathcal{N}$

output:

- The index $\tilde{k}_i$ associated to $\bar{y}$

1 $\tilde{k}_i \leftarrow 0$
2 $r \leftarrow 1$
3 $val \leftarrow 1 - \bar{y}_{\mathcal{S}_{ir}}$
4 **while** $val > 0$ *and* $r < M$ **do**
5 **if** $d_{i(\mathcal{S}_{i(r+1)})} > d_{i\mathcal{S}_{ir}}$ **then**
6 $\tilde{k}_i \leftarrow \tilde{k}_i + 1$
7 $r \leftarrow (r + 1)$
8 $val \leftarrow val - \bar{y}_{\mathcal{S}_{ir}}$
9 **return** $\tilde{k}_i$

3.3.4 Benders reformulation

The Benders cuts (3.20) lead to the following formulation for (PMP):

$$
(F4)\begin{cases}
\min & \displaystyle\sum_{i\in\mathcal{N}}\theta_i & & \\
\text{s.t.} & \displaystyle\sum_{j\in\mathcal{M}} y_j = p & & \\
& \theta_i \geq D_i^1 & i\in\mathcal{N} & (3.21)\\
& \theta_i \geq D_i^{k+1} - \displaystyle\sum_{j:d_{ij}\leq D_i^k}(D_i^{k+1}-d_{ij})y_j & i\in\mathcal{N},\ k\in\{1,...,K_i-1\} & (3.22)\\
& y_j \in \{0,1\} & j\in\mathcal{M} &
\end{cases}
$$

Constraints (3.22) indicate that each allocation distance θ_i is D_i^{k+1} unless a site is opened at a distance less than or equal to D_i^k from i. This formulation $(F4)$ has $(N+M)$ variables which is less than $(F2)$ and $(F3)$ but it has the same number of constraints. Nevertheless, the constraint matrix is roughly as dense as $(F2)$ and it has the same continuous relaxation value.

Table 3.1 presents the results of four formulations of the (PMP) on five instances from OR-Library described in Section 3.4.1. A time limit of 600 seconds is considered. For each formulation, we present the relative optimality gap (gap) and the runtime in seconds ($t[s]$).

INSTANCE				F1		F2		F3		F4	
name	$N=M$	p	OPT	gap	$t[s]$	gap	$t[s]$	gap	$t[s]$	gap	$t[s]$
pmed26	600	5	9917	0%	228	0%	40	0%	**8**	0%	57
pmed31	700	5	10,086	0%	282	0%	36	0%	**7**	0%	58
pmed35	800	5	10,400	0%	527	0%	104	0%	**9**	0%	95
pmed38	900	5	11,060	74.1%	600	0%	75	0%	**19**	0%	115
pmed39	900	5	11,069	10.7%	600	0%	66	0%	**19**	0%	105
pmed40	900	5	12,305	0%	579	0%	60	0%	**10**	0%	104

Table 3.1: Comparison between different (PMP) formulations with a time limit of 600 seconds.

Results in Table 3.1 confirm the expected performance between formulations $(F1)$, $(F2)$ and $(F3)$ already described in Section 3.2.1. Moreover, $(F4)$ takes more time than $(F2)$ and $(F3)$.

3.3.5 Decomposition algorithm implementation

To improve the performance of the Benders decomposition, a two-phase algorithm is implemented. Let $(\overline{MP})$ be the master problem without the integrality constraints. The Benders decomposition is first solved for $(\overline{MP})$ (*Phase 1*). Then, the integrality constraints are added to the obtained master problem to solve it through a *branch-and-cut* algorithm (*Phase 2*).

Phase 1: Solving the linear relaxation of the master problem

Phase 1 is summarized in Algorithm 4. The current master problem $\overline{MP}$ is solved at Step 4 through a linear programming solver and provides a candidate solution $(\bar{y}, \bar{\theta})$ while the sub-problems are solved at Steps 2 and 6 using Algorithm 2. To enhance the performance this phase includes the following improvements:

- **Initial solution**: Providing a good candidate solution to the initial $(\overline{MP})$ can significantly reduce the number of iterations. Consequently, as García et al. (2011), a first solution is computed using the `PopStar` heuristic (Resende and Werneck (2004)) which, to the best of our knowledge, is the best heuristic for the (PMP). `PopStar` is a hybrid heuristic that combines elements of several metaheuristics. It uses a multi-start method in which a solution is built at each iteration as in a GRASP algorithm. It is followed by an intensification strategy, with a tabu search and a scatter search. And in a post-optimization phase, they use the concept of multiple generations, a characteristic of genetic algorithms. The solution y^h provided by this heuristic and its objective value UB^h are inputs of Algorithm 4. The latter is used to initialize UB^1 at Step 1.

- **Rounding heuristic**: Since in Phase 1 most of the solutions provided by $(\overline{MP})$ are fractional, we use a primal heuristic to try to improve the upper bound of the problem. At each iteration, the sites associated to the p largest values of $\bar{y}$ are opened (Steps 7 to 11 in Algorithm 4).

The objective value of $(\overline{MP})$ and the sub-problem optimal value $OPT(SP)$ allow us to update the optimality bounds on the value of the linear relaxation of the problem. In each iteration the rounding heuristic tries the improve UB^1. The iterative algorithm is terminated when no more violated Benders cuts are found for the current solution $\overline{y}$ and hence the value of the linear relaxation of the problem is obtained.

Algorithm 4: Phase 1 - Solving $(\overline{MP})$

input :

- Instance data (N, M, p , Distances $D_i^1, ..., D_i^{K_i}$ and d_{ij} with $i \in \mathcal{N}$, $j \in \mathcal{M}$)
- Heuristic (PMP) solution y^h with value UB^h

output:

- Lower bound LB^1 and a feasible integer solution y^1 with value UB^1

$(y^1, UB^1) \leftarrow (y^h, UB^h)$

Use Algorithm 1 to generate violated Benders cuts associated with y^h to $(\overline{MP})$

while *violated cut has been found* **do**

  $(\bar{y}, \bar{\theta}) \leftarrow$ Solve $(\overline{MP})$

  $LB^1 \leftarrow \sum_{i=1}^{N} \bar{\theta}_i$

  Use Algorithm 1 to generate violated Benders cuts associated with $\bar{y}$ to $(\overline{MP})$

  if $\bar{y}$ *is fractional* **then**

    $(y^r, UB^r) \leftarrow$ Get a rounded heuristic solution from $\bar{y}$

    if $UB^r < UB^1$ **then**

      $UB^1 \leftarrow UB^r$

      $y^1 \leftarrow y^r$

return LB^1, y^1, UB^1

Phase 2: Solving the master problem with *branch-and-Benders-cut* approach

Once the continuous relaxation of (MP) is solved by Phase 1, the integrality constraints on variables y are added and a *branch-and-cut* algorithm is used to solve the problem. The sub-problems are solved at each node which provides an integer solution in order to generate Benders cuts. The solution of the sub-problems is performed through callbacks which is a feature provided by mixed integer programming solvers. In order to enhance the performance of Phase 2, the following improvements are inplemented:

- **Constraint reduction**: At the end of Phase 1, most of the generated Benders cuts are not saturated by the current fractional solution. Most of them are removed to reduce the problem size. The cuts of a client $i \in \mathcal{N}$ are related to indexes $\tilde{k}_i$ obtained at different iterations. Let $\hat{k}$ be the highest of these indexes associated with a saturated constraint. All constraints of client $i \in \mathcal{N}$ which associated index is higher than $\hat{k}$ are removed. This reduction performs better than removing all unsaturated constraints.

- **Reduced cost fixing**: At the end of Phase 1, given the bounds LB^1 and UB^1, it is possible to perform an analysis of the reduced costs $\overline{rc}$ of the last fractional solution $\overline{y}$ provided by Algorithm 4. For any site $j \in \mathcal{M}$ such that $LB^1 + \overline{rc}_j > UB^1$, y_j can be set to 0. Similarly, for any site $j \in \mathcal{M}$ such that $LB^1 - \overline{rc}_j > UB^1$, y_j can be fixed to 1. It is computationally observed that that these rules are efficient in instances where p is small, i.e., when the ratio p/M is less than 20%. At higher values of p, there may exist many equivalent solutions. Therefore, opening or closing a site often does not have a strong impact on the objective.

3.4 Computational study

In this section, the results of the Benders decomposition method with those of the state-of-the-art exact methods described in Section 3.2.2 are compared.

3.4.1 Benchmark instances

We study the same instances used in García et al. (2011) that is the p-median instances from OR-Library (Beasley (1990)) and TSP-Library (Reinelt (1991)). In all these instances, the sites are at the same location as the clients and thus $N = M$. The set of OR-Library contains instances with 100 to 900 clients, and the value of p is between 5 and 500. The set of TSP-Library selected contains between 1304 and 238,025 clients. Following previous works such as García et al. (2011), all client points are given as two-dimensional coordinates, and the Euclidean distance rounded down to the nearest integer is used as distance.

Another set of symmetric instances that satisfy triangle inequality are the BIRCH instances, usually solved by heuristics algorithms (see e.g Hansen et al. (2009); Avella et al. (2012); Irawan et al. (2014)). These instances consist of p clusters of two-dimensional data points generated in a square. We consider instances with sizes from 10,000 to 20,000 points and from 25,000 to 89,600 points for two types of instances, named Type I and Type III. These instances were kindly provided by the authors of Avella et al. (2012). For the comparison, we consider the results presented in Avella et al. (2012), in which an aggregation heuristic is proposed. This heuristic is denoted as `AvellaHeu`. The results presented in Irawan and Salhi (2015b) are considered for large BIRCH instances, in which a hybrid heuristic combining aggregation and variable neighborhood search is proposed. This heuristic is denoted as `IrawanHeu`.

We also consider the RW instances originally proposed by Resende and Werneck (2004) with the `PopStar` heuristic. They correspond to completely random distance matrices. The distance between each center and each customer is an integer value taken uniformly between the values 1

to 1000. Moreover, the distance between client i and site j is not necessarily equal to the distance between site j and client i. Four different values of $N = M$ are considered: 100, 250, 500, and 1000.

Finally, we include in the experimentation the ODM instances which were introduced by Briant and Naddef (2004) and are used in Avella et al. (2007) with a *branch-and-cut-and-price* algorithm. This algorithm is denoted as `AvellaB&C`. These instances correspond to the *optimal diversity management problem* which can be treated as a p-median problem in which certain allocations between clients and sites are not allowed. For this problem there exist instances with N equal to 1284, 3773, and 5335. However, we only consider the hardest instances with $N = 3773$.

3.4.2 Technical specifications

Our study was carried out on an Intel XEON W-2145 processor 3.7 GHz, with 16 threads, but only 1 was used, and 256 GB of RAM. IBM ILOG CPLEX 20.1 was used as *branch-and-cut* framework. The described separation algorithm is applied in the GenericCallback of CPLEX, which gets called whenever a feasible integer solution is found. The absolute tolerance to the best integer objective (EpGap) has been set to 10^{-10}, and the tolerance to the best remaining node, also called absolute MIP gap (EpAGap), to 0.9999.

Considering that the Benders decomposition can easily find feasible solutions, the MIP emphasis switch has been set to BestBound in order to prove the optimality as fast as possible. The branch-up-first parameter BRDIR has been set to 1, since this tends to produce branching trees with fewer nodes. It used a time limit of 10 hours for Phase 2, indicated by TL in the tables when this is reached.

We run the `Zebra` and `PopStar` methods on our computer. `Zebra` code was provided by the authors of García et al. (2011) and `PopStar` code is available online[1]. On the other hand, we do not report an updated time for the heuristic algorithms: `AvellaB&C` was originally carried out on a Compaq EVO W4000 Personal Computer with Pentium IV-1.8 GHz processor with 1 GB of RAM using the LP solver IBM ILOG CPLEX 8.0 with a time limit of 100 hours per instance (indicated by TL2 in the corresponding table), `AvellaHeu` was carried out on an Intel Core 2 Quad CPU 2.6 GHz workstation with 4 GB of RAM with a single core, and `IrawanHeu` was carried out on a PC Intel Core i5 CPU 650@ 3.20 GHz of processor with 4 GB of RAM. In order to compare more objectively the computation times of our method with the ones of these approaches, we consider the benchmark score of each computer from the geekbench website[2]. To

[1] http://mauricio.resende.info/popstar/
[2] https://browser.geekbench.com

obtain a solution time for these three methods closer to the one that would have been obtained if we had executed them, one can multiply the time reported in their articles by the ratio between the score of the used computer and the score of the computer on which they were obtained (i.e., 3 for `AvellaHeu`, 2.5 for `IrawanHeu`, and 6 for `AvellaB&C`).

3.4.3 Performance analysis

The results for the different instances are presented below. The information in the tables is organized as follows:

- Instance data:
  - *name*: name of the instance.
  - $N = M$: size of the instance (number of clients equal to the number of sites).
  - p: number of sites to open.
  - OPT/BKN: optimal value of the instance (in **bold**) if it is known or the best-known solution value obtained given the time limit, otherwise. If the value is underlined, it means that it is the first time the instance is solved to optimality or that the best-known value was improved.
- Our Phase 1 results:
  - LB^1: lower bound of the (PMP) obtained at the end of Phase 1.
  - UB^1: upper bound of the (PMP) obtained at the end of Phase 1.
  - $t^1[s]$: CPU time in seconds required to complete Phase 1.
- Our Phase 1 + Phase 2 results:
  - *gap*: relative optimality gap between the lower and upper bound obtained in Phase 2.
  - *iter*: number of total iterations required for the Benders decomposition, i.e., the number of times a fractional solution or an integer solution was separated in Phase 1 and Phase 2, respectively.
  - *nodes*: number of the explored nodes of the *branch-and-cut*.
  - $t^{tot}[s]$: the total CPU time in seconds required to exactly solve the instance.
- `Zebra`, `AvellaHeu`, `IrawanHeu`, `PopStar`, and `AvellaB&C` results:
  - *gap* / UB^h: relative optimality gap when available or the solution value obtained at the end of the corresponding method.

  - $t[s]$: total CPU time time in seconds required to complete the algorithms of García et al. (2011), Avella et al. (2012), Irawan and Salhi (2015b), Resende and Werneck (2004) or Avella et al. (2007) respectively. A diamond (⧫) means that the computer ran out of memory while solving the problem.

- Average total time:
  - the average total time by our method and `Zebra` is presented in the corresponding tables. This average is calculated considering only the instances where both methods solve the instances to optimality.

OR-Library and TSP-Library instances

We present the results on OR-Library in Table 3.2 and on TSP-Library instances in Tables 3.3, 3.4, 3.5, and 3.6 for small, medium, large, and huge instances, respectively. Similarly to `Zebra`, the optimal value of all the OR-Library instances is reached in few seconds. For the TSP instances, our method reaches the optimal solution in most instances. Very good LB^1 and UB^1 bounds are quickly found at the end of Phase 1.

In Tables 3.3 and 3.4, it is possible to observe that 10 small and medium instances are not solved optimally by `Zebra` due to a lack of memory or for reaching the time limit. However, our method does not face any memory problem and only 2 small and 2 medium instances reach the time limit of 10 hours with an optimality gap lower than $0,1\%$.

INSTANCE				PHASE 1			PHASE 1 + 2			Zebra	
name	$N = M$	p	OPT/BKN	LB^1	UB^1	$t^1[s]$	*gap*	*iter*	$t^{tot}[s]$	*gap*	$t[s]$
pmed26	600	5	**9917**	9854	9917	0.14	0%	15	1.12	0%	5.58
pmed27	600	10	**8306**	8302	8307	0.18	0%	19	0.63	0%	0.91
pmed28	600	60	**4498**	4498	4498	0.12	0%	7	0.16	0%	0.12
pmed29	600	120	**3033**	3033	3033	0.08	0%	8	0.11	0%	0.05
pmed30	600	200	**1989**	1989	1989	0.04	0%	9	0.04	0%	0.04
pmed31	700	5	**10,086**	10,026	10,086	0.16	0%	13	0.93	0%	4.23
pmed32	700	10	**9297**	9293	9297	0.27	0%	9	0.91	0%	1.29
pmed33	700	70	**4700**	4700	4700	0.18	0%	8	0.24	0%	0.17
pmed34	700	140	**3013**	3013	3013	0.08	0%	7	0.08	0%	0.07
pmed35	800	5	**10,400**	10,302	10,400	0.17	0%	9	1.24	0%	5.32
pmed36	800	10	**9934**	9834	9934	0.25	0%	18	27.2	0%	26.7
pmed37	800	80	**5057**	5057	5057	0.22	0%	6	0.29	0%	0.15
pmed38	900	5	**11,060**	10,948	11,060	0.23	0%	15	4.42	0%	10.1
pmed38	900	10	**9431**	9362	9431	0.31	0%	20	4.17	0%	10.3
pmed38	900	20	**7839**	7832	7839	0.32	0%	11	1.03	0%	2.87
pmed38	900	50	**5892**	5889	5892	0.26	0%	11	0.86	0%	1.19
pmed38	900	100	**4450**	4450	4450	0.22	0%	8	0.31	0%	0.15
pmed38	900	200	**2905**	2905	2905	0.09	0%	8	0.09	0%	0.08
pmed38	900	300	**1972**	1972	1972	0.07	0%	8	0.07	0%	0.06
pmed38	900	400	**1305**	1305	1305	0.07	0%	10	0.07	0%	0.06
pmed38	900	500	**836**	836	836	0.05	0%	8	0.08	0%	0.05
pmed39	900	5	**11,069**	10,938	11,069	0.29	0%	22	3.66	0%	8.61
pmed39	900	10	**9423**	9365	9423	0.31	0%	18	6.27	0%	8.01
pmed39	900	20	**7894**	7894	7894	0.34	0%	9	0.52	0%	0.72
pmed39	900	50	**5941**	5937	5941	0.33	0%	12	0.95	0%	1.21
pmed39	900	100	**4461**	4461	4462	0.24	0%	9	0.33	0%	0.36
pmed39	900	200	**2918**	2918	2918	0.09	0%	7	0.14	0%	0.08
pmed39	900	300	**1968**	1968	1968	0.07	0%	7	0.11	0%	0.06
pmed39	900	400	**1303**	1303	1303	0.08	0%	10	0.08	0%	0.06
pmed39	900	500	**821**	821	821	0.05	0%	8	0.08	0%	0.05
pmed40	900	5	**12,305**	12,246	12,305	0.27	0%	11	1.41	0%	3.87
pmed40	900	10	**10,491**	10,439	10,491	0.33	0%	27	2.77	0%	5.54
pmed40	900	20	**8717**	8711	8717	0.39	0%	13	1.27	0%	2.39
pmed40	900	50	**6518**	6505	6518	0.33	0%	11	2.94	0%	4.15
pmed40	900	90	**5128**	5128	5128	0.23	0%	8	0.23	0%	0.24
pmed40	900	200	**3132**	3132	3132	0.11	0%	8	0.11	0%	0.09
pmed40	900	300	**2106**	2106	2106	0.09	0%	11	0.09	0%	0.07
pmed40	900	400	**1398**	1398	1398	0.06	0%	9	0.06	0%	0.06
pmed40	900	500	**900**	900	900	0.04	0%	7	0.07	0%	0.05
					Average total time				1.67		2.69

Table 3.2: Results on ORLIB instances for our method and `Zebra` on our computer.

INSTANCE				PHASE 1			PHASE 1 + 2				Zebra	
name	$N = M$	p	OPT/BKN	LB^1	UB^1	$t^1[s]$	*gap*	*iter*	*nodes*	$t^{tot}[s]$	*gap*	$t[s]$
rl1304	1304	5	**3,099,073**	3,099,073	3,099,073	2.70	0%	9	0	2.8	0%	1233
		10	**2,134,295**	2,131,788	2,134,295	2.90	0%	12	160	15.4	0%	1060
		20	**1,412,108**	1,412,108	1,412,108	2.25	0%	8	0	2.3	0%	61
		50	**795,012**	795,012	795,012	1.46	0%	9	0	1.5	0%	8.3
		100	**491,639**	491,507	491,788	0.90	0%	19	37	2.4	0%	3.6
		200	**268,573**	268,573	268,573	0.35	0%	11	0	0.5	0%	0.9
		300	**177,326**	177,318	177,339	0.31	0%	12	0	0.5	0%	0.5
		400	**128,332**	128,332	128,332	0.23	0%	10	0	0.2	0%	0.1
		500	**97,024**	97,018	97,034	0.27	0%	14	0	0.4	0%	0.2
fl1400	1400	5	**174,877**	174,877	174,877	0.82	0%	7	0	0.9	0%	245
		10	**100,601**	100,601	100,601	0.40	0%	6	0	0.4	0%	72
		20	**57,191**	57,191	57,191	0.38	0%	8	0	0.4	0%	10
		50	**28,486**	28,486	28,486	0.36	0%	8	0	0.4	0%	2.5
		100	**15,962**	15,961	15,962	0.82	0%	12	5	2.1	0%	5.0
		200	**8806**	8793	8815	0.66	0%	20	570	26.9	0%	305
		300	**6109**	6092	6157	0.76	0%	28	12,599	385	9%	TL
		400	**4648**	4636	4659	0.56	0%	51	6,716,041	32,655	8%	TL
		500	3764	3756	3773	0.53	0.09%	44	4,987,858	TL	8%	TL
u1432	1432	5	**1,210,126**	1,210,126	1210126	1.65	0%	7	0	1.8	0%	324
		10	**849,759**	849,759	849,759	3.47	0%	7	0	3.5	0%	71
		20	**588,766**	588,720	588,767	3.86	0%	12	3	5.8	0%	18
		50	**362,072**	361,724	362,072	4.28	0%	25	1493	161	0%	128
		100	**243,793**	243,758	243,850	1.86	0%	12	0	3.0	0%	7.3
		200	**159,887**	159,867	160,084	0.72	0%	13	8	2.0	0%	1.9
		300	**123,689**	123,674	123,876	0.63	0%	15	0	1.0	0%	2.2
		400	103,979	103,411	104,102	0.91	0.11%	29	5,806,518	TL	0%	TL
		500	**93,200**	93,200	93,200	0.16	0%	8	0	0.3	0%	0.1
vm1748	1748	5	**4,479,421**	4,479,421	4,479,421	2.36	0%	7	0	2.5	0%	4955
		10	**2,983,645**	2,983,048	2,983,645	4.67	0%	14	11	10.9	0%	1364
		20	**1,899,680**	1,899,588	1,899,681	4.76	0%	15	5	9.5	0%	309
		50	**1,004,331**	1,004,325	1,004,339	2.23	0%	12	0	2.8	0%	21
		100	**636,515**	636,418	636,541	1.57	0%	15	3	3.2	0%	13
		200	**390,350**	390,350	390,350	0.79	0%	11	0	0.8	0%	1.6
		300	**286,039**	286,037	286,080	0.63	0%	13	0	1.0	0%	1.0
		400	**221,526**	221,523	221,545	0.53	0%	13	0	0.8	0%	1.0
		500	**176,986**	176,977	177,103	0.54	0%	14	0	0.8	0%	0.5
					Average total time					9		320

Table 3.3: Results on small TSP instances for our method and `Zebra` on our computer. TL=36,000 seconds. The average total time is computed on the instances solved to optimality by both approaches.

INSTANCE				PHASE 1			PHASE 1 + 2				Zebra	
name	$N = M$	p	OPT/BKN	LB^1	UB^1	$t^1[s]$	*gap*	*iter*	*nodes*	$t^{tot}[s]$	*gap*	$t[s]$
d2103	2103	5	**1,005,136**	1,005,136	1,005,136	4	0%	8	0	4	0%	4268
		10	**687,321**	687,264	687,321	8	0%	11	7	12	0%	1085
		20	**482,926**	482,798	482,926	9	0%	12	71	18	0%	448
		50	302,219	301,592	302,219	18	0.04%	34	149,711	TL	0%	7203
		100	**194,664**	194,408	194,994	17	0%	39	95,993	10,363	0%	16,022
		200	**117,753**	117,736	117,778	2	0%	16	3	5	0%	5
		300	**90,471**	90,424	90,510	2	0%	21	0	3	0%	29
		400	**75,324**	75,291	75,425	1	0%	19	2	4	0%	6209
		500	**64,006**	63,952	64,315	1	0%	27	477	8	0%	2568
pcb3038	3038	5	**1,777,835**	1,777,665	1,777,835	29	0%	12	13	55	0%	TL
		10	**1,211,704**	1,211,704	1,211,704	18	0%	8	0	18	0%	19,526
		20	**839,494**	839,233	839,499	50	0%	18	188	146	0%	7329
		50	**506,339**	506,205	506,339	24	0%	12	194	85	0%	1134
		100	**351,500**	351,404	351,648	28	0%	22	390	96	0%	346
		150	**280,128**	280,058	280,423	16	0%	24	640	269	0%	148
		200	**237,399**	237,328	237,578	11	0%	13	734	84	0%	130
		300	**186,833**	186,793	186,906	6	0%	19	73	17	0%	60
		400	**156,276**	156,268	156,307	3	0%	10	0	6	0%	22
		500	**134,798**	134,774	134,866	2	0%	17	0	4	0%	17
fl3795	3795	5	**1,052,627**	1,052,627	1,052,627	14	0%	7	0	14	∞	⧫
		10	**520,940**	520,940	520,940	8	0%	8	0	8	0%	4410
		20	**319,722**	319,722	319,722	6	0%	11	0	6	0%	2671
		50	**150,940**	150,940	150,940	5	0%	13	0	5	0%	193
		100	**88,299**	88,299	88,299	6	0%	12	0	6	0%	45
		150	**65,868**	65,840	65,904	14	0%	52	1833	221	0%	1825
		200	**53,928**	53,913	54,013	11	0%	68	46,730	2633	0%	2501
		300	**39,586**	39,578	39,661	6	0%	47	127,800	2548	0%	3061
		400	**31,354**	31,348	31,472	5	0%	57	14,566	559	0%	527
		500	**25,976**	25,976	25,976	6	0%	15	0	6	0%	2
rl5934	5934	10	**9,792,218**	9,786,688	9,792,218	405	0%	17	329	1864	∞	⧫
		20	**6,716,215**	6,713,214	6,716,228	437	0%	27	1381	14,725	∞	⧫
		50	4,029,999	4,026,936	4,029,999	362	0.03%	32	8068	TL	0%	TL
		200	**1,805,530**	1,805,030	1,807,763	67	0%	27	1353	967	0%	3816
		300	**1,392,419**	1,392,304	1,392,709	38	0%	15	25	58	0%	235
		400	**1,143,940**	1,143,649	1,145,342	20	0%	33	1809	303	0%	1110
		500	**972,799**	972,741	973,712	17	0%	20	117	44	0%	70
		600	**847,301**	847,233	847,769	12	0%	16	0	18	0%	77
		700	**751,131**	751,054	751,569	7	0%	15	0	14	0%	88
		800	**675,958**	675,884	676,248	7	0%	20	175	21	0%	58
		900	**612,629**	612,574	612,879	7	0%	17	35	14	0%	33
		1000	**558,167**	558,088	558,311	7	0%	28	603	45	0%	731
		1100	**511,192**	511,138	511,453	7	0%	22	43	15	0%	20
		1200	**469,747**	469,712	469,943	8	0%	18	0	11	0%	11
		1300	**433,060**	433,015	433,300	7	0%	19	5	12	0%	27
		1400	**401,370**	401,356	401,597	7	0%	15	0	9	0%	6
		1500	**373,566**	373,566	373,566	7	0%	17	0	7	0%	2
					Average total time					467		2022

Table 3.4: Results on medium TSP instances for our method and `Zebra` on our computer. TL=36,000 seconds. ⧫ means that the computer ran out of memory. The average total time is computed on the instances solved to optimality by both approaches.

Regarding the large and huge instances in Tables 3.5 and 3.6, 57 of the 68 instances are solved, whereas `Zebra` only solves 16 instances due to a lack of memory. For the huge instances, the rounding heuristic step of Phase 1, the reduced cost fixing step, and the constraint reduction step of Phase 2 are not used as they take too much time. Nevertheless, the rounding heuristic is used once at the end of Phase 1 to update UB^1. Moreover, for the huge instances, a randomly generated solution is used instead of `PopStar` which takes a long time. For the first time the optimal values for 7 instances with $N = M = 115,455$ and 10 instances with $N = M = 238,025$ are provided.

INSTANCE				PHASE 1			PHASE 1 + 2				Zebra	
name	$N = M$	p	$OPT/$ BKN	LB^1	UB^1	$t^1[s]$	*gap*	*iter*	*nodes*	$t^{tot}[s]$	*gap*	$t[s]$
usa13509	13,509	10	**398,561,730**	398,561,600	398,561,730	288	0%	10	0	755	∞	⧫
		25	**234,600,221**	234,600,221	234,600,221	455	0%	10	0	455	∞	⧫
		50	**157,819,849**	157,815,657	157,819,849	431	0%	11	45	646	∞	⧫
		100	**108,002,205**	107,983,102	108,002,411	523	0%	23	605	4043	∞	⧫
		200	**74,220,726**	74,213,328	74,229,411	426	0%	25	969	2269	∞	⧫
		300	**59,340,915**	59,334,913	59,346,783	473	0%	23	984	1744	0%	18,760
		400	**50,538,905**	50,533,013	50,575,463	319	0%	24	874	2556	0%	23,677
		500	**44,469,860**	44,463,038	44,499,566	278	0%	36	2883	3945	0%	25,105
		600	**39,952,138**	39,944,049	39,991,088	295	0%	34	49,175	23,712	0%	TL
		700	**36,469,603**	36,463,603	36,512,930	202	0%	24	6060	2551	0%	TL
		800	**33,635,127**	33,631,192	33,672,848	210	0%	21	796	1215	0%	6007
		900	**31,275,114**	31,269,089	31,299,760	182	0%	31	27,272	11,851	0%	TL
		1000	**29,268,216**	29,262,339	29,309,009	154	0%	31	942	1382	0%	TL
		2000	**18,230,856**	18,229,432	18,238,229	47	0%	21	202	125	0%	584
		3000	**13,098,935**	13,097,929	13,101,469	49	0%	28	9	72	0%	1674
		4000	**9,905,715**	9,905,071	9,910,848	37	0%	17	0	50	0%	166
		5000	**7,608,605**	7,608,242	7,611,958	45	0%	22	0	61	0%	86
sw24978	24,978	10	**22,670,073**	22,670,073	22,670,073	1037	0%	12	0	1037	∞	⧫
		25	**14,085,626**	1,4085,352	14,085,626	6651	0%	11	15	9447	∞	⧫
		50	**9,652,817**	9,652,817	9,652,817	4136	0%	10	0	4136	∞	⧫
		75	7,766,486	7,765,106	7,766,486	6310	0.010%	12	457	TL	∞	⧫
		100	6,660,424	6,657,806	6,660,424	9634	0.031%	14	228	TL	∞	⧫
		250	4,034,554	4,034,055	4,036,558	2413	0.003%	14	699	TL	∞	⧫
		500	2,747,215	2,746,498	2,751,695	1866	0.015%	20	2621	TL	∞	⧫
		1000	**1,841,723**	1,841,613	1,844,801	621	0%	23	1061	3814	0%	30,796
		2000	**1,197,278**	1,197,231	1,198,464	208	0%	17	132	476	0%	TL
		3000	**911,361**	911,308	911,988	145	0%	17	16	344	0%	3614
		4000	**737,645**	737,602	738,045	92	0%	15	0	190	0%	TL
		5000	**617,637**	617,593	618,096	76	0%	18	0	127	0%	TL
		6000	**527,336**	527,307	527,716	72	0%	17	0	112	0%	5188
		7000	**455,716**	455,696	456,074	63	0%	16	0	99	0%	3973
		8000	**397,217**	397,153	397,540	44	0%	20	0	97	0%	TL
		9000	**347,376**	347,322	347,621	44	0%	20	13	92	0%	TL
		10,000	**305,998**	305,932	306,094	33	0%	17	0	60	0%	TL
					Average total time					1178		9969

Table 3.5: Results on large TSP instances for our method and `Zebra` on our computer. TL=36,000 seconds. ⧫ means that the computer ran out of memory. The average total time is computed on the instances solved to optimality by both approaches.

INSTANCE				PHASE 1			PHASE 1 + 2				Zebra	
name	$N = M$	p	OPT/BKN	LB^1	UB^1	$t^1[s]$	*gap*	*iter*	*nodes*	$t^{tot}[s]$	*gap*	$t[s]$
ch71009	71,009	10,000	4,274,662	4,273,680	4,424,131	6326	0.006%	36	18,585	TL	∞	⧫
		20,000	**2,377,760**	2,377,409	2,419,539	681	0%	40	474	3581	∞	⧫
		30,000	**1,464,151**	1,464,015	1,473,517	431	0%	27	0	819	∞	⧫
		40,000	**879,336**	879,272	881,997	220	0%	17	0	465	∞	⧫
		50,000	**463,553**	463,544	463,904	133	0%	24	0	258	0%	653
		60,000	**167,565**	167,558	167,789	49	0%	31	0	135	0%	331
pla85900	85,900	10,000	166,853,134	166,627,292	182,428,500	2841	0.12%	30	2113	TL	∞	⧫
		20,000	109,007,210	107,246,411	120,645,337	3975	1.58%	27	618	TL	∞	⧫
		30,000	86,944,862	86,944,715	87,547,287	1411	0.0002%	84	28,033	TL	∞	⧫
		40,000	**69,944,715**	69,944,715	69,965,668	1006	0%	12	0	1006	∞	⧫
		50,000	**52,944,715**	52,944,715	52,945,623	921	0%	12	0	921	∞	⧫
		60,000	**35,944,715**	35,944,715	35,945,105	858	0%	11	0	858	∞	⧫
		70,000	**18,977,475**	18,977,475	18,977,475	73	0%	13	0	73	0%	122
		80,000	**4,512,752**	4,512,752	4,512,752	12	0%	20	0	13	0%	97
usa115475	115,474	20,000	5,287,343	5,286,659	5,383,798	3366	0.001%	36	11102	TL	∞	⧫
		30,000	**3,815,620**	3,815,143	3,861,590	1494	0%	41	589	11,581	∞	⧫
		40,000	**2,876,909**	2,876,603	290,4492	1353	0%	32	459	4431	∞	⧫
		50,000	**2,189,144**	2,188,903	2,200,969	1122	0%	28	480	3189	∞	⧫
		60,000	**1,651,400**	1,651,234	1,657,118	795	0%	25	0	1588	∞	⧫
		70,000	**1,214,299**	1,214,177	1,217,251	612	0%	17	0	1045	∞	⧫
		80,000	**851,481**	851,422	852,851	435	0%	24	0	788	∞	⧫
		90,000	**548,097**	548,076	548,560	270	0%	18	0	544	∞	⧫
ara238025	238,025	10,000	1354,335	1,345,698	1,446,100	5197	0.64%	19	0	TL	∞	⧫
		20,000	857,553	857,453	878,372	5582	0.004%	42	696	TL	∞	⧫
		30,000	**630,969**	630,872	643,171	5123	0%	33	663	33,687	∞	⧫
		40,000	**494,842**	494,804	498,378	4135	0%	18	0	10,028	∞	⧫
		50,000	**401,835**	401,795	404,218	2675	0%	19	0	8327	∞	⧫
		60,000	**334,279**	334,236	335,807	2969	0%	17	0	7240	∞	⧫
		70,000	**283,627**	283,592	286,065	3058	0%	28	509	19,298	∞	⧫
		80,000	**244,233**	243,936	248,742	2578	0%	36	378	14,615	∞	⧫
		90,000	**214,233**	213,936	219673	1548	0%	27	507	28391	∞	⧫
		1000,00	**184,233**	184,069	188,556	1973	0%	44	613	18,473	∞	⧫
		150,000	**88,025**	88,025	88,334	1532	0%	27	0	6057	∞	⧫
		200,000	**38,025**	38,025	38,025	319	0%	11	0	319	∞	⧫
					Average total time					120		300

Table 3.6: Results on huge TSP instances for our method and `Zebra` on our computer. TL=36,000 seconds. ⧫ means that the computer ran out of memory. The average total time is computed on the instances solved to optimality by both approaches.

BIRCH instances

The results on BIRCH instances are summarized in Tables 3.7 and 3.8. Almost all of these instances were solved in the first phase either by finding an integer solution directly or by the rounding heuristic. Consequently, optimal values are quickly obtained for all these instances. Even when multiplying the solution time of `AvellaHeu` and `IrawanHeu` by their benchmark ratio (2.5 and 3, respectively), our approach remains the best for most instances.

INSTANCE				PHASE 1			PHASE 1 + 2				AvellaHeu	
name	$N = M$	p	OPT/BKN	LB^1	UB^1	$t^1[s]$	*gap*	*iter*	*nodes*	$t^{tot}[s]$	UB^h	$t[s]$
ds1x1	10,000	100	**12,428.5**	12,428.5	12,428.5	9	0%	6	0	9	**12,428.5**	47
ds1x2	15,000	100	**18,639.3**	18,639.3	18,639.3	29	0%	7	0	29	**18,639.3**	101
ds1x3	20,000	100	**24,840.3**	24,840.3	24,840.3	38	0%	7	0	38	**24,840.3**	210
ds1x4	9600	64	**11,934.8**	11,934.8	11,934.8	13	0%	6	0	13	**11,934.8**	56
ds1x5	12,800	64	**15,863.8**	15,863.8	15,863.8	25	0%	7	0	25	**15,863.8**	84
ds1x6	16,000	64	**20,004.5**	20,004.5	20,004.5	31	0%	6	0	31	**20,004.6**	129
ds1x7	19,200	64	**24,018.3**	24,018.3	24,018.3	55	0%	6	0	55	**24,018.3**	219
ds1x8	10,000	25	**12,455.7**	12,455.7	12,455.7	28	0%	6	0	28	**12,455.7**	82
ds1x9	12,500	25	**15,597.1**	15,597.1	15,597.1	43	0%	6	0	43	**15,597.1**	115
ds1x0	15,000	25	**18,949.3**	18,949.3	18,949.3	67	0%	7	0	67	**18,949.3**	175
ds1xA	17,500	25	**21,937.4**	21,937.4	21,937.4	116	0%	6	0	116	**21,937.4**	241
ds1xB	20,000	25	**25,096.8**	25,096.8	25,096.8	108	0%	6	0	108	**25,096.8**	365
ds3x1	10,000	100	**9624.8**	9624.8	9624.8	16	0%	9	0	16	**9624.8**	60
ds3x2	15,000	100	**15,898.2**	15,895.9	15,899.1	39	0%	10	88	142	15,904.1	121
ds3x3	20,000	100	**19,976.2**	19,974.6	19,977.6	97	0%	10	13	260	19,989.0	222
ds3x4	9600	64	**8225.6**	8224.1	8225.7	24	0%	12	5	65	**8225.6**	57
ds3x5	12,800	64	**10,210.4**	10,210.4	10,210.4	36	0%	10	0	36	**10,210.4**	98
ds3x6	16,000	64	**13,335.4**	13,335.4	13,335.4	62	0%	10	0	62	13,340.5	170
ds3x7	19,200	64	**15,207.6**	15,207.1	15,207.6	176	0%	18	0	400	**15,207.6**	229
ds3x8	10,000	25	**7203.4**	7203.4	7203.4	61	0%	11	0	61	**7203.4**	94
ds3x9	12,500	25	**8576.1**	8576.1	8576.1	68	0%	7	0	68	**8576.1**	144
ds3x0	15,000	25	**9513.6**	9513.6	9513.6	135	0%	13	0	136	**9513.6**	192
ds3xA	17,500	25	**12,535.7**	12,535.7	12,535.7	224	0%	16	0	224	**12,535.7**	250
ds3xB	20,000	25	**13,022.2**	13,022.2	13,022.2	244	0%	11	0	244	13,052.8	364

Table 3.7: Results on BIRCH instances for our method and the results of `AvellaHeu` reported in Avella et al. (2012).

INSTANCE				PHASE 1			PHASE 1 + 2				IrawanHeu	
name	$N = M$	p	OPT/BKN	LB^1	UB^1	$t^1[s]$	*gap*	*iter*	*nodes*	$t^{tot}[s]$	UB^h	$t[s]$
ds1n01	25,000	25	**31,229.4**	31,229.4	31,229.4	153	0%	7	0	153	31,282.6	447
ds1n02	36,000	36	**45,115.6**	45,115.6	45,115.6	311	0%	7	0	311	45,115.6	780
ds1n03	49,000	49	**61,384.1**	61,384.1	61,384.1	388	0%	7	0	388	61,569.7	1216
ds1n04	64,000	64	**80,053.9**	80,053.9	80,053.9	675	0%	7	0	675	80,377.4	2258
ds1n05	30,000	25	**37,563.6**	37,563.6	37,563.6	305	0%	7	0	305	37,617.1	559
ds1n06	43,200	36	**54,191.4**	54,191.4	54,191.4	320	0%	7	0	320	54,305.8	1003
ds1n07	58,800	49	**73,626.8**	73,626.8	73,626.8	683	0%	7	0	683	73,854.7	1691
ds1n08	76,800	64	**96,039.4**	96,039.4	96,039.4	949	0%	7	0	949	96,393.4	2834
ds1n09	35,000	25	**43,902.1**	43,902.1	43,902.1	385	0%	7	0	386	42,972.1	758
ds1n10	50,400	36	**63,169.2**	63,169.2	63,169.2	533	0%	7	0	533	63,329.2	1472
ds1n11	68,600	49	**85,833.5**	85,833.5	85,833.5	910	0%	8	0	910	86,082.0	2441
ds1n12	89,600	64	**112,059.2**	112,059.2	112,059.2	1332	0%	7	0	1332	112,485.2	4501
ds3n01	25,000	25	**17,696.2**	17,696.2	17,696.2	112	0%	6	0	112	17,718.6	527
ds3n02	36,000	36	**27,423.0**	27,423.0	27,423.0	237	0%	7	0	237	27,476.1	913
ds3n03	49,000	49	**44,149.0**	44,149.0	44,149.0	294	0%	10	0	295	44,282.5	1760
ds3n04	64,000	64	**58,832.6**	58,832.6	58,832.6	807	0%	11	0	807	58,991.5	2624
ds3n05	30,000	25	**21,829.9**	21,829.9	21,829.9	258	0%	7	0	258	21,865.1	832
ds3n06	43,200	36	**32,339.4**	32,339.4	32,339.4	337	0%	9	0	337	32,391.6	1873
ds3n07	58,800	49	**50,857.9**	50,857.9	50,857.9	831	0%	12	0	831	50,857.9	2692
ds3n08	76,800	64	**66,561.4**	66,561.0	66,655.7	1490	0%	17	9	4587	66,944.7	4393
ds3n09	35,000	25	**24,810.9**	24,810.9	24,810.9	869	0%	10	0	869	24,833.7	972
ds3n10	50,400	36	**38,102.6**	38,102.6	38,102.6	504	0%	8	0	504	38,162.3	2297
ds3n11	68,600	49	**61,850.6**	61,850.6	61,850.6	1065	0%	14	0	1065	62,007.4	2556
ds3n12	89,600	64	**78,777.0**	78,777.0	78,777.0	1548	0%	19	0	1548	79,245.3	5779

Table 3.8: Results on large BIRCH instances for our method and the results of `IrawanHeu` reported in Irawan and Salhi (2015b).

RW instances

The results on RW instances are summarized in Table 3.9. Even small RW instances can be very difficult to solve as previously observed by Elloumi and Plateau (2010). This would be mainly due to the fact that the instances are not Euclidean. Furthermore, the total number of distances K is closer to $N \times M$, leading to more variables and constraints in Formulations $(F2)$ and $(F3)$. For large values of p, our decomposition can quickly solve the instances to optimality. These instances were not considered by either Avella et al. (2007) or García et al. (2011). Moreover, the code of `Zebra` cannot handle non-symmetric instances. Consequently, it is only reported the computation time and value UB^h of the solution computed by the heuristic `PopStar`.

INSTANCE				PHASE 1			PHASE 1 + 2				PopStar	
name	$N = M$	p	OPT	LB^1	UB^1	$t^1[s]$	*gap*	*iter*	*nodes*	$t^{tot}[s]$	UB^h	$t[s]$
rw100_10	100	10	**530**	475	530	0.02	0%	46	4817	6.82	**530**	0.05
rw100_20	100	20	**277**	274	277	0.01	0%	9	0	0.20	**277**	0.03
rw100_30	100	30	**213**	213	213	0.01	0%	9	0	0.01	**213**	0.02
rw100_40	100	40	**187**	187	187	0.01	0%	7	0	0.01	**187**	0.03
rw100_50	100	50	**172**	172	172	0.01	0%	7	0	0.01	**172**	0.02
rw250_10	250	10	**3691**	2811	3698	0.26	0%	71	4,072,666	31,099	3698	0.31
rw250_25	250	25	**1360**	1216	1364	0.14	0.4%	64	5,003,894	TL	1364	0.34
rw250_50	250	50	**713**	699	713	0.07	0%	23	2050	7.31	**713**	0.15
rw250_75	250	75	**523**	523	523	0.02	0%	11	0	0.02	524	0.09
rw250_100	250	100	**444**	444	444	0.02	0%	9	0	0.02	**444**	0.08
rw250_125	250	125	**411**	411	411	0.02	0%	8	0	0.02	**411**	0.07
rw500_10	500	10	16,108	11,012	16,144	1.35	21.5%	81	211,046	TL	16,144	2.53
rw500_25	500	25	5683	4403	5716	0.85	16.3%	73	506,777	TL	5716	1.80
rw500_50	500	50	2627	2321	2627	0.52	6.5%	44	1,290,135	TL	2627	1.03
rw500_75	500	75	1757	1672	1757	0.36	1.1%	31	2,436,992	TL	1757	0.84
rw500_100	500	100	**1379**	1353	1382	0.25	0%	45	20,9745	1482	1382	0.47
rw500_150	500	150	**1024**	1024	1024	0.05	0%	8	0	0.05	**1024**	0.30
rw500_250	500	250	**833**	833	833	0.03	0%	9	0	0.03	**833**	0.25
rw1000_10	1000	10	68,136	44,697	68,136	10.35	36.2%	61	19,577	TL	68,136	8.65
rw1000_25	1000	25	24,964	17,387	25,042	6.22	34.1%	77	32,697	TL	25,042	7.60
rw1000_50	1000	50	11,328	8760	11,328	5.18	23.2%	65	94,037	TL	11.328	7.09
rw1000_75	1000	75	7207	5998	7223	4.20	16.2%	70	151,511	TL	7223	3.15
rw1000_100	1000	100	5233	4631	5233	3.22	9.7%	42	285,579	TL	5233	4.45
rw1000_200	1000	200	2710	2664	2710	0.94	0.5%	31	1,310,025	TL	2710	3.27
rw1000_300	1000	300	**2018**	2017	2018	0.20	0%	12	0	0.28	**2018**	1.99
rw1000_400	1000	400	**1734**	1734	1734	0.08	0%	9	0	0.09	**1734**	1.68
rw1000_500	1000	500	**1614**	1614	1614	0.07	0%	8	0	0.08	**1614**	1.35

Table 3.9: Results on RW instances for our exact method and `PopStar` heuristic in our computer. TL=36,000 seconds.

ODM instances

The results on ODM instances are summarized in Table 3.10. To solve these instances, N constraints have to be added to ensure that each client is allocated to one of its non-forbidden neighbors. This generates a more complex master problem to solve. The rounding heuristic could not be used directly with these instances, so in order to save computation time, it was not used.

Obtaining a solution for these instances is hard since `PopStar` does not support their format and since random solutions may not be feasible due to the sparsity of the graphs. Consequently, to obtain an initial solution, its corresponding formulation $(F3)$ by CPLEX is solved and stopped it once it has found 3 feasible solutions. This approach empirically proved to be a good compromise between computation time and optimality gap. It was not possible to be able to use `Zebra` for these instances. All these instances were solved to optimality. Our results remains the best on all instances even after multiplying `AvellaB&C` solution times by its benchmark ratio of value 6.

INSTANCE				PHASE 1			PHASE 1 + 2				AvellaB&C	
name	$N = M$	p	OPT	LB^1	UB^1	$t^1[s]$	*gap*	*iter*	*nodes*	$t^{tot}[s]$	*gap*	$t[s]$
BD3773	3773	5	**726,954,998.4**	715,785,543.5	748,669,824.0	6	0%	123	59	58	0%	1540
		6	**685,812,258.0**	673,317,265.9	720,632,505.6	10	0%	318	221	158	0%	41,551
		7	**651,930,471.0**	636,565,701.5	727,428,978.0	11	0%	1165	627	589	0%	216,851
		8	**620,886,605.4**	606,599,816.1	1,157,543,276.4	20	0%	2520	1153	1434	1.7%	329,053
		9	**595,955,799.0**	581,022,150.3	649,313,415.0	18	0%	4028	1981	2385	2.6%	TL2
		10	**574,634,206.8**	559,096,162.3	633,383,307.0	24	0%	8229	4848	4372	2.8%	TL2
		11	**554,972,029.2**	539,810,124.3	603,741,870.0	29	0%	10,139	5940	5832	2.9%	TL2
		12	**536,700,087.0**	522,614,063.7	605,415,767.4	30	0%	13,951	5898	8083	3.1%	TL2
		13	**521,375,065.2**	507,136,693.1	581,851,884.6	31	0%	22,500	9471	12,581	3.2%	TL2
		14	**507,756,740.4**	493,051,932.7	550,267,457.4	35	0%	52,705	30,275	31,651	3.1%	TL2

Table 3.10: Results on ODM instances for our method and the results of `AvellaB&C` reported in Avella et al. (2007). TL2=360000 seconds.

3.4.4 Adaptation for the Uncapacitaded Facility Location problem

Given the closeness of the (PMP) and the (UFL), we compare our two-phase decomposition algorithm with the approach proposed in Fischetti et al. (2017) for the (UFL). They solve the problem in a *branch-and-cut* approach in which they search for violated Benders cuts for both the integer solutions and the fractional solutions of the linear relaxations with a ad-hoc algorithm. This approach is also known as *branch-and-Benders-cut*. They also consider some stabilization techniques and heuristics for the cut loop at the root and at the branching nodes. Their method is denoted as `BBC`.

We consider the same set of KG instances from UFLLIB[3] to compare the performance on the linear formulation. These instances can be divided into three groups, with $N = M \in \{250, 500, 750\}$. Within each KG group, there are two classes of instances, symmetric and asymmetric ones, denoted by "gs" and "ga", respectively. Additionally, each class contains three sub-classes, "a", "b," and "c," representing different cost settings: in subclass a, allocation costs are an order of magnitude higher than the facility opening costs; in subclass b, these costs are of the same order; and in subclass c, facility opening costs are an order of magnitude higher than the allocation costs.

The computational study in Fischetti et al. (2017) was conducted on a cluster of identical machines each consisting of an Intel Xeon E3-1220V2 CPU running at 3.10 GHz, with 16 GB of RAM each. They reported the wall-clock times and referred to four-thread runs with a time limit of 2 hours per instance (indicated by TL3 in the corresponding table).

KG Instances

The results on KG instances are summarized in 3.11. To solve these instances, the master problem needs to be modified to consider the open cost of sites and remove the constraint of limiting the number of open sites to p. This generates a harder master problem to solve. Let c_j be the cost of to open the site $j \in \mathcal{M}$.

$$(MP_{UFL}) \begin{cases} \min & \sum_{i \in \mathcal{N}} \theta_i + \sum_{j \in \mathcal{M}} c_j y_j & \\ \text{s.t.} & \sum_{j \in \mathcal{M}} y_j = p & \\ & \theta_i \text{ satisfies } BD_i & i \in \mathcal{N} \\ & y_j \in \{0,1\} & j \in \mathcal{M} \end{cases}$$

The heuristic used in Phase 1 must also be modified, since PopStar does not take into account site opening costs and the p parameter is not available. A greedy heuristic is considered to obtain the initial solution. As a rounding heuristic, all the sites which $\overline{y}_j > 0.4$ for $j \in \mathcal{M}$ are set to 1.

The obtained results are consistent with those presented in Fischetti et al. (2017) since within the 2-hour limit it is also not possible to optimally solve the considered instances. Fischetti et al. (2017) do not report lower bounds on the optimal value. It is observed here that the final gap is relatively small, as its maximum value is 1.7%. However, Fischetti et al. (2017) have a better upper bound for all instances which is probably due to the stabilization technique and its primal heuristic used at each node.

[3] https://resources.mpi-inf.mpg.de/departments/d1/projects/benchmarks/UflLib/index.html

INSTANCE			PHASE 1				PHASE 1 + 2				BBC	
name	$N = M$	BKN	LB^1	UB^1	$t^1[s]$	UB^2	*gap*	*iter*	*nodes*	$t^{tot}[s]$	UB^h	$t[s]$
ga500a-1	500	511,383	510,589.0	514,040	2.0	511,474	0.11%	65	227,655	TL3	511,383	TL3
ga500a-2	500	511,255	510,472.2	514,397	1.8	511,367	0.11%	69	217,453	TL3	511,255	TL3
ga500a-3	500	510,810	510,139.0	513,356	1.9	510,965	0.09%	78	264,644	TL3	510,810	TL3
ga500a-4	500	511,008	510,382.6	512,694	1.7	511,082	0.08%	35	369,649	TL3	511,008	TL3
ga500a-5	500	511,239	510,487.7	513,475	2.0	511,425	0.11%	74	237,491	TL3	511,239	TL3
ga500b-1	500	538,060	533,338.6	545,628	1.6	538,452	0.49%	66	119,621	TL3	538,060	TL3
ga500b-2	500	537,850	533,087.6	619,389	1.6	538,457	0.60%	84	87,311	TL3	537,850	TL3
ga500b-3	500	537,924	532,735.7	626,127	1.7	538,264	0.63%	77	76,708	TL3	537,924	TL3
ga500b-4	500	537,925	532,841.9	670,237	1.5	538,385	0.58%	90	86,797	TL3	537,925	TL3
ga500b-5	500	537,482	532,968.8	597,360	1.5	537,662	0.44%	71	124,594	TL3	537,482	TL3
gs500a-1	500	511,188	510,409.9	513,472	3.4	511,314	0.10%	95	233,059	TL3	511,188	TL3
gs500a-2	500	511,179	510,448.7	514,006	1.9	511,354	0.11%	82	165,089	TL3	511,179	TL3
gs500a-3	500	511,112	510,321.4	513,779	2.3	511,287	0.12%	85	141,758	TL3	511,112	TL3
gs500a-4	500	511,137	510,369.6	513,787	2.2	511,278	0.11%	66	159,787	TL3	511,137	TL3
gs500a-5	500	511,293	510,494.5	513,990	2.0	511,532	0.13%	83	145,791	TL3	511,293	TL3
gs500b-1	500	537,931	533,026.7	659,721	1.4	538,418	0.57%	76	84,363	TL3	537,931	TL3
gs500b-2	500	537,763	533,096.1	662,841	1.4	538,160	0.51%	67	113,638	TL3	537,763	TL3
gs500b-3	500	537,854	532,832.2	678,990	1.7	538,457	0.63%	91	76,097	TL3	537,854	TL3
gs500b-4	500	537,742	532,717.2	664,753	1.5	538,422	0.66%	97	74,407	TL3	537,742	TL3
gs500b-5	500	538,270	533,098.2	669,016	1.6	538,618	0.59%	80	83,721	TL3	538,270	TL3
ga750a-1	750	763,528	762,464.5	766,540	5.8	763,869	0.15%	93	50,507	TL3	763,528	TL3
ga750a-2	750	762,653	762,520.2	767,067	9.2	763,973	0.15%	89	46,602	TL3	762,653	TL3
ga750a-3	750	763,697	762,568.5	766,608	6.2	763,930	0.14%	79	43,302	TL3	763,697	TL3
ga750a-4	750	763,945	762,738.5	767,839	6.6	764,240	0.16%	77	36,036	TL3	763,945	TL3
ga750a-5	750	763,786	762,637.0	767,096	7.3	764,159	0.16%	75	40,176	TL3	763,786	TL3
ga750b-1	750	796,454	790,121.9	897,053	4.6	797,090	0.62%	66	26,275	TL3	796,454	TL3
ga750b-2	750	795,963	789,512.4	938,670	4.5	796,498	0.62%	82	25,485	TL3	795,963	TL3
ga750b-3	750	796,130	789,618.5	929,140	4.6	796,640	0.63%	79	23,499	TL3	796,359	TL3
ga750b-4	750	797,013	790,345.1	926,574	4.7	797,935	0.69%	90	22,494	TL3	797,013	TL3
ga750b-5	750	796,387	789,647.3	930,851	4.0	796,934	0.66%	89	23,036	TL3	796,549	TL3
ga750c-1	750	902,026	875,624.7	1,018,182	2.2	903,292	1.59%	87	23,967	TL3	902,026	TL3
ga750c-2	750	899,651	873,946.7	1,017,899	2.8	902,368	1.70%	84	22,481	TL3	899,651	TL3
ga750c-3	750	900,010	874,108.9	1,012,861	3.1	902,099	1.66%	82	24,265	TL3	900,019	TL3
ga750c-4	750	900,044	875,565.9	1,028,964	2.4	901,809	1.42%	76	23,248	TL3	900,044	TL3
ga750c-5	750	899,235	873,191.5	1,028,543	2.6	900,541	1.55%	87	24,812	TL3	899,235	TL3
gs750a-1	750	763,671	762,564.9	767,232	6.5	763,925	0.14%	84	43,430	TL3	763,671	TL3
gs750a-2	750	763,548	762,529.4	766,414	6.1	763,666	0.11%	90	57,546	TL3	763,548	TL3
gs750a-3	750	763,727	762,568.1	765,552	6.6	764,031	0.16%	82	41,309	TL3	763,727	TL3
gs750a-4	750	763,887	762,788.3	766,768	7.5	764,208	0.15%	81	36,651	TL3	763,922	TL3
gs750a-5	750	763,614	762,528.9	766,741	6.4	763,947	0.15%	92	44,013	TL3	763,614	TL3
gs750b-1	750	797,026	790,349.4	994,689	4.7	797,713	0.68%	82	20,525	TL3	797,329	TL3
gs750b-2	750	796,170	789,669.9	895,766	4.1	796,843	0.66%	80	26,652	TL3	796,170	TL3
gs750b-3	750	796,589	789,935.6	996,019	4.5	797,357	0.69%	84	22,153	TL3	796,589	TL3
gs750b-4	750	796,734	790,080.8	997,279	4.2	797,176	0.65%	69	26,574	TL3	797,020	TL3
gs750b-5	750	796,365	789,902.8	930,041	4.6	797,026	0.63%	63	23,510	TL3	796,365	TL3
gs750c-1	750	900,363	875,363.9	1,020,684	2.9	903,801	1.64%	80	22,196	TL3	900,363	TL3
gs750c-2	750	897,886	874,186.6	1,004,896	2.5	900,668	1.49%	70	27,230	TL3	897,886	TL3
gs750c-3	750	901,656	874,762.5	1,019,498	2.5	902,767	1.61%	89	21,335	TL3	901,656	TL3
gs750c-4	750	901,239	875,410.6	1,007,402	2.6	902,591	1.61%	81	24,195	TL3	901,239	TL3
gs750c-5	750	900,216	875,956.4	1,017,315	3.4	903,719	1.65%	78	20,022	TL3	900,216	TL3

Table 3.11: Results on KG instances for our method and the results of `BBC` reported in Fischetti et al. (2017) TL3=72,000 seconds.

3.5 Conclusions

We define a Benders decomposition based on the most efficient formulation of the p-median problem. The efficiency of our decomposition comes from a fast algorithm for the solution of the sub-problems in conjunction with additional improvements in a two-phase method. In the first phase, the integrality constraints are relaxed and in the second phase, the problem is solved in an efficient *branch-and-cut* approach.

Our approach outperforms other state-of-the-art methods. For the first time, instances are solved having up to 89,600 and 238,025 clients and sites from the BIRCH and TSP libraries, respectively. Our decomposition method is tested on other p-median instances: RW instances which do not satisfy triangle inequality and ODM instances in which there are allocations that are not allowed between certain clients and sites. For the RW instances, we solve instances of up to 1000 clients with a large value of p. For ODM instances, we solve previously unsolved instances with 3773 nodes within 10 hours. Our approach was also adapted to test it on the difficult KG instances of the (UFL) problem obtaining relatively small optimality gaps.

One of the perspectives of this research is to exploit these results on other families of location problems. It is also expected to use other branching strategies that allow a greater efficiency during the development of the *branch-and-cut* algorithm.

Chapter 4

New Benders decomposition and Clustering algorithm for the p-Center Problem

4.1 Introduction

The *p-center problem* (PCP) is also a fundamental problem in location science (Laporte et al. (2019b)). It consists of choosing the p locations to minimize the maximum distance from a client to its nearest facility. The p open facilities are called *centers*. Formally the problem is defined as follows. We consider a set of N clients, a set of M potential facilities to open, and their corresponding index set $\mathcal{N} = \{1, ..., N\}$ and $\mathcal{M} = \{1, ..., M\}$. Let d_{ij} be the distance between client i and center j, and p be an integer. The objective is to find a set $S \subseteq \mathcal{M}$ such that $|S| \leq p$ and the value $z = \max_{i \in \mathcal{N}} (\min_{j \in S} d_{ij})$ is minimized. This distance z is called the *radius*. Figure 4.1 illustrates a comparison of (PCP) solutions for different values of p considering the same symmetric instance used in the introduction (Chapter 1). Note how the corresponding radius, shown in red, decreases as p increases.

The p-center problem in operations research is applied in various contexts, including facility location, emergency service planning, telecommunications network design, healthcare planning, and supply chain management. It allows to decide the best places for centers, such as warehouses or hospitals, to minimize distances or response times (Çalık et al. (2019a)).

There are a variety of exact and approximate methods for this classical problem and its variants. In recent years, important advances have been made in the solution of large-scale instances. To the best of our knowledge, the best methods are those of Contardo et al. (2019) and Gaar and

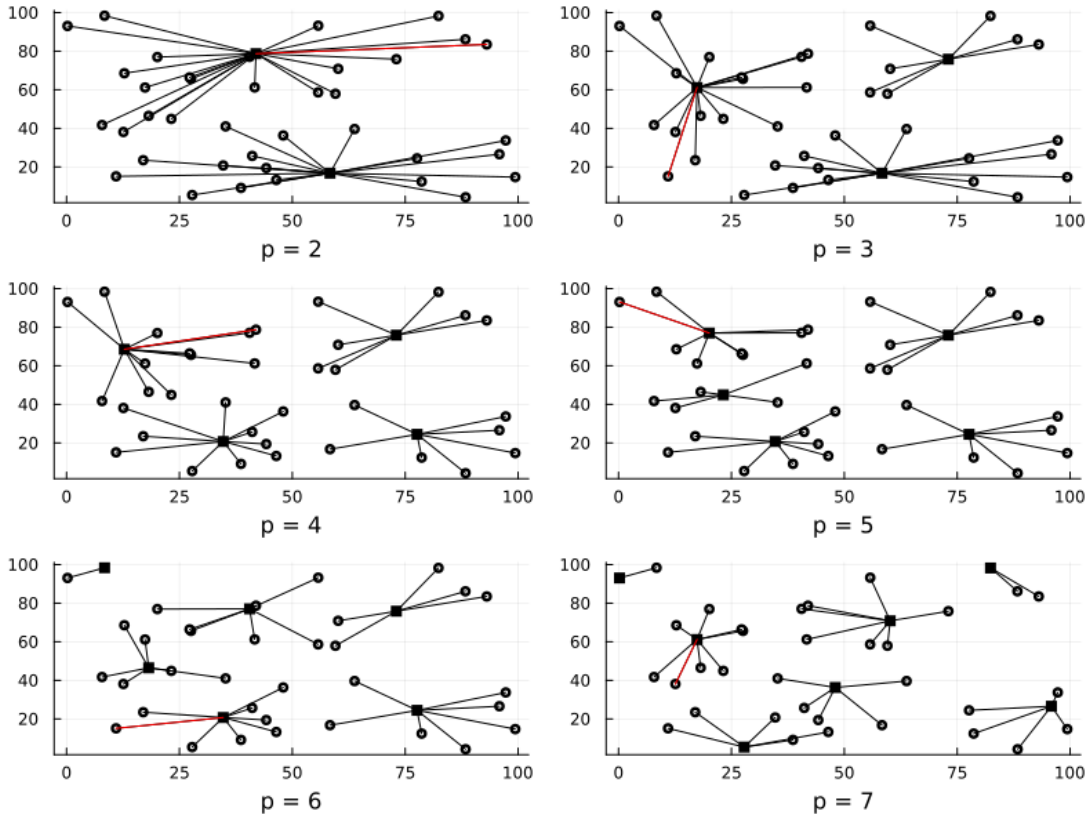


Figure 4.1: Comparison of optimal PCP solutions for a symmetric instance of 40 nodes.

Sinnl (2022). The former initially considers a subset of clients, solves the (PCP) associated to this subset, and adds new clients until an optimal solution is obtained. The latter method is based on a specialized branch-and-cut procedure for a Benders decomposition. Both methods can solve very large-scale instances.

Contribution

Firstly, we explore two new Benders decompositions for the (PCP). Secondly, we introduce an exact algorithm based on a clustering procedure. Finally, we present the comparative results with the state-of-the-art exact methods for the (PCP) on benchmark instances.

The rest of the chapter is organized as follows: Section 4.2 presents the literature review of the (PCP). In Section 4.3, we present a comparison of performances of the main MILP formulations. In Section 4.4 and Section 4.5, two new Benders decomposition of the same (PCP) formulation are studied. Section 4.6 presents our new exact solution method based on a clustering of the clients. The results of the computational study are presented in Section 4.7. Finally, some conclusions together with research perspectives are drawn in Section 4.8.

4.2 Literature review

The p-center problem was introduced by Hakimi (1965), who presented and solved the absolute 1-center problem on a graph. In the *absolute* p-center problem, the centers can be located either on the edges or the vertices of the graph. Later, Minieka (1970) extended the problem to the case $p > 1$ and proposed a method to restrict the continuous set of candidate centers to a discrete set of points, without losing optimality. Since then, the problem was commonly referred to as the *vertex* p-center problem or directly as the p-center problem. Several formulations, solution methods, and variants of this problem have been presented. We refer to Çalık et al. (2019b) for a more exhaustive review of applications and solution methods for the p-center problem. The following is a summary of the main MILP formulations and the state-of-the-art exact solution methods.

4.2.1 Mathematical formulations

The problem can be formulated with a binary variable y_j for each center j which is 1 if center j is opened, 0 otherwise; and another binary variable x_{ij} which is 1 if client i is allocated to center j, 0 otherwise.

$$(F0)\begin{cases} \min \max\limits_{i\in\mathcal{N}} & \sum\limits_{j\in\mathcal{M}} d_{ij}x_{ij} & & (4.1)\\ \text{s.t.} & \sum\limits_{j\in\mathcal{M}} y_j = p & & (4.2)\\ & \sum\limits_{j\in\mathcal{M}} x_{ij} = 1 & i\in\mathcal{N} & (4.3)\\ & x_{ij} \le y_j & i\in\mathcal{N}, j\in\mathcal{M} & (4.4)\\ & x_{ij}, y_j \in \{0,1\} & i\in\mathcal{N}, j\in\mathcal{M} & (4.5) \end{cases}$$

Constraint (4.2) limits the number of open facilities to p. Constraints (4.3) ensures that each client is allocated to only one center and Constraints (4.4) ensures that no client is allocated to a center that is not open.

The formulation proposed in Daskin (1996) is considered as the classical formulation, which treats the objective *min-max* function by introducing a new real variable z representing the maximum distance between a client and its nearest center, the *radius*.

$$
(F1)\begin{cases}
\min & z & & (4.6)\\
\text{s.t.} & \sum_{j\in\mathcal{M}} y_j = p & & (4.7)\\
& \sum_{j\in\mathcal{M}} x_{ij} = 1 & i\in\mathcal{N} & (4.8)\\
& x_{ij} \le y_j & i\in\mathcal{N}, j\in\mathcal{M} & (4.9)\\
& \sum_{j\in\mathcal{M}} d_{ij}x_{ij} \le z & i\in\mathcal{N} & (4.10)\\
& x_{ij}, y_j \in \{0,1\} & i\in\mathcal{N}, j\in\mathcal{M} & (4.11)\\
& z\in\mathbb{R} & & (4.12)
\end{cases}
$$

Constraints (4.7), (4.8), and (4.9) come from the first model. Then the objective function is reformulated and Constraints (4.10) are added to indicate that the distances between each client and its nearest center are less than the radius.

Elloumi et al. (2004) proposed an alternative formulation which orders all distinct values of the distances between each pair of clients and centers. Let $D^0 < D^1 < ... < D^K$ be the distinct values of d_{ij} for all $i \in \mathcal{N}$ and $j \in \mathcal{M}$. Let $\mathcal{K} = \{1, ..., K\}$ be the respective distance index set without considering the smallest distance. The radius variable z and the allocation variables x are replaced by variables z^k with $k \in \mathcal{K}$ equal to 1 if the radius is greater than or equal to D^k. The *initial* formulation they obtain is as follows:

$$
(F2_{init})\begin{cases}
\min & D^0 + \sum_{k\in\mathcal{K}} (D^k - D^{k-1}) z^k & & (4.13)\\
\text{s.t.} & \sum_{j\in\mathcal{M}} y_j = p & & (4.14)\\
& z^k + \sum_{j:d_{ij}<D^k} y_j \ge 1 & i\in\mathcal{N}, k\in\mathcal{K} & (4.15)\\
& y_j \in \{0,1\} & j\in\mathcal{M} & (4.16)\\
& z^k \in \{0,1\} & k\in\mathcal{K} & (4.17)
\end{cases}
$$

Constraint (4.14) limit the number of open facilities to p. Constraints (4.15) indicates that a client is covered by a center at a distance less than D^k, or that the radius is greater than or equal to D^k. Thus, if $z^k = 1$, $(D^k - D^{k-1})$ is then added in Objective (4.13).

Both formulations are polynomial in size but have different structures. The classical model has $(N+1) \times M$ binary variables, 1 continuous variable, and $N \times (M+2) + 1$ constraints. $(F2)$ has $(M+K)$ binary variables and $(K \times N + 1)$ constraints. In the worst case where K is close to

$(N \times M)$, the two problems have almost the same number of variables, but $(F2_{init})$ has about N times as many constraints. Elloumi et al. (2004) show that $(F2_{init})$ provides a continuous relaxation bound that dominates that of $(F1)$.

Later, Ales and Elloumi (2018) present a formulation that improves the resolution performance of the previous formulation $(F2_{init})$. They add the following family of valid inequalities:

$$z^k \geq z^{k+1} \qquad k \in \{1, ..., K-1\}$$

and they also identify the following sub-set of indices:

$$\mathcal{K}_i = \{k \in \mathcal{K} : \exists\, j \in \mathcal{M} \text{ such that } d_{ij} = D^k\} \qquad i \in \mathcal{N}$$

Consequently, they propose the following formulation:

$$(F2) \begin{cases} \min & D^0 + \sum_{k \in \mathcal{K}} (D^k - D^{k-1}) z^k & & \\ \text{s.t.} & z^k + \sum_{j : d_{ij} < D^k} y_j \geq 1 & i \in \mathcal{N}, k \in \mathcal{K}_i & (4.18) \\ & z^k \geq z^{k+1} & k \in \{1, 2, \ldots, K-1\} & (4.19) \\ & \sum_{j \in \mathcal{M}} y_j = p & & \\ & y_j \in \{0, 1\} & j \in \mathcal{M} & \\ & z^k \in \{0, 1\} & k \in \mathcal{K} & \end{cases}$$

Constraints (4.19) allow to remove a significant number of Constraints (4.15). Constraints (4.18) represent the subset of (4.15) that are not redundant. Indeed, when there is no site j such that $d_{ij} = D^k$, Constraint (4.15) for i and k is dominated by Constraint (4.15) for i and $k+1$ and can therefore be omitted. It is proved in Ales and Elloumi (2018) that even if $(F2)$ is much lighter than $(F2_{init})$, it has the same continuous relaxation bound.

Ales and Elloumi (2018) also presented another compact formulation, which contains less variables and constraints than $(F2)$. They replace the K binary variables z_k with a unique integer variable r which represents the index of the optimal radius:

$$
(F3)\begin{cases}
\min & r & & (4.20)\\
\text{s.t.} & r \geq k - k \sum\limits_{j:d_{ij}<D^k} y_j & i \in \mathcal{N}, k \in \mathcal{K}_i & (4.21)\\
& \sum\limits_{j\in\mathcal{M}} y_j = p & &\\
& y_j \in \{0,1\} & j \in \mathcal{M} &\\
& r \geq 0 & &
\end{cases}
$$

Constraints (4.21) play a similar role to that of Constraints (4.18). This formulation $(F3)$ provides a weaker linear relaxation than the previous formulations.

Calik and Tansel (2013) introduce two mathematical formulations. Here we consider the tightest one and denote it as $(F4)$. They consider the same binary variables y_j of $(F1)$ and a binary variable u^k for all $k \in \{0, ..., K\}$ which is equal to 1 if and only if the optimal radius is equal to D^k. Thus, exactly one of these new binary variables is equal to one. The solutions of these formulations can be mapped to those of $(F2_{init})$ by setting $u^0 = 1 - z^1$, $u^k = z^k - z^{k+1}$, and $z^k = 1 - \sum\limits_{k=0}^{K} u^k$.

$$
(F4)\begin{cases}
\min & \sum\limits_{k=0}^{K} D^k u^k & & (4.22)\\
\text{s.t.} & \sum\limits_{j\in\mathcal{M}\,:\,d_{ij}\leq D^k} y_j \geq \sum\limits_{q=0}^{k} u^q & i \in \mathcal{N}, k \in \{0, ..., K\} & (4.23)\\
& \sum\limits_{k=0}^{K} u^k = 1 & & (4.24)\\
& \sum\limits_{j\in\mathcal{M}} y_j = p & & (4.25)\\
& y_j \in \{0,1\} & j \in \mathcal{M} & (4.26)\\
& u^k \in \{0,1\} & k \in \{0, ..., K\} & (4.27)
\end{cases}
$$

Finally, Gaar and Sinnl (2022) recently presented a formulation obtained from a Benders decomposition of $(F1)$ which is closely related to a formulation of the *uncapacitated facility location problem* (UFL) from Cornuejols et al. (1980) and Magnanti and Wong (1981). This formulation of (UFL) also considers binary variables x from $(F1)$ and a single continuous variable θ to represent the radius which is minimized in the objective.

$$(F5)\begin{cases} \min & \theta & & (4.28) \\ \text{s.t.} & \sum_{j\in\mathcal{M}} y_j = p & & \\ & \theta \geq d_{ij} - \sum_{j':d_{ij'}<d_{ij}} (d_{ij} - d_{ij'})y_{j'} & i \in \mathcal{N}, j \in \mathcal{M} & (4.29) \\ & y_j \in \{0,1\} & j \in \mathcal{M} & \\ & \theta \in \mathbb{R} & & \end{cases}$$

Constraints (4.29) ensure that the radius is at least the distance to the nearest center for each demand node. These constraints can be seen as a reinforcement of the Constraints (4.21) in $(F3)$.

4.2.2 Solutions methods

Several heuristic and exact methods have been proposed for (PCP). In this research we are interested in the exact solution methods. We refer to the survey Garcia-Diaz et al. (2019) for approximation algorithms and heuristics. The following is a summary of the main exact solution methods since the introduction of (PCP).

The first exact solution method was presented in Minieka (1970), it is proposed an iterative algorithm in which at each step, a set cover problem is solved to determine if there exists a solution of radius at most equal to a distance in $D = \{D^0, D^1, ..., D^K\}$. Garfinkel et al. (1977) introduce an heuristic as initial solution and a binary search for the remaining distances in D. T. Ilhan (2002) adopt the approach of Minieka (1970) proposing a first phase to obtain the bounds and thus to reduce the size of D to the second stage with the set cover problems.

Later, Al-Khedhairi (2005) improve the second stage with a sequential search in order to reduce the number of iterations. Caruso et al. (2003) have proposed exact and heuristic algorithms based on the idea of strong and weak dominance rules. Daskin (1996) adopted the same binary search idea, but solved maximum coverage problems instead of set cover problems. Elloumi et al. (2004) proposes some modifications to the algorithm of Daskin (1996) calculating a continuous solution of the set-cover problems at each iteration instead of one optimal solution based in their proposed formulation $(F2_{init})$. R. Chen (1987) and Chen and Chen (2009) presented iterative algorithms based on relaxation that consider only subsets of clients. If the greatest dissimilarity between a client that is not in the subset and its nearest center in the solution of the relaxed problem is less than or equal to the value of the solution, then this solution is also feasible (and therefore optimal) for the original instance. If this is not the case, the subset is updated by adding a constant number of clients.

More recently, Calik and Tansel (2013) developed a search that solves a restricted model containing only two blocks of coverage constraints based in their proposed formulation $(F4)$. Contardo et al. (2019) have proposed to solve the problem considering only a sub-set of clients and updating this set when it is needed. It also considers an improved binary search with the set cover problems. Finally, Gaar and Sinnl (2022) proposed a scalable method based on a specialized branch-and-cut procedure for the Benders decomposition on $(F1)$.

Several variants of the PCP have been introduced and investigated in the literature, such as the incorporation of capacities, demands or uncertainty in the parameters. We refer to Çalık et al. (2019a) for more details. Since the most efficient methods use decomposition algorithms that take advantage of lower bounds. In Sections 4.4 and 4.5, we introduce two new Benders decompositions. Moreover, we present an efficient algorithm based on client clustering in Section 4.6.

4.3 Performance comparison of the MILP formulations

In this section we compare the efficiency of the direct solution of the five formulations presented in Section 4.2.1 by a standard MILP solver. For this, we use the default setting of CPLEX 20.1 with a time limit of 2 hours. We solve the 40 symmetric ORLIB instances (Beasley (1990)) of 100 to 900 nodes. The optimal values and their corresponding CPU times are presented in Table 4.1.

We observe that $(F1)$ is the only formulation that cannot solve all instances reaching the time limit. $(F4)$ is the fastest formulation on average followed by $(F2)$ and $(F5)$. Nevertheless, the best results are almost always obtained either by $(F2)$ or $(F4)$.

However, the performance of the formulations can be influenced by the different parameters of the MILP solver, among these an important one is the pre-solve parameter. In Table 4.2 we present the results obtained on the same instances but without CPLEX pre-solve and with a time limit of 30 minutes. $(F2)$ is now the only formulation able to solve all the instances. $(F4)$ is the second best which only reaches the time limit for 5 instances. These performance differences are mainly due to the fact that the pre-solve can eliminate many redundant constraints.

Instance				t[s]				
name	$N = M$	p	OPT	F1	F2	F3	F4	F5
pmed01	100	5	127	13	**3**	10	25	15
pmed02	100	10	98	8	**2**	16	25	15
pmed03	100	10	93	7	**3**	18	26	19
pmed04	100	20	74	4	**2**	194	25	20
pmed05	100	33	48	**2**	**2**	11	21	16
pmed06	200	5	84	106	**18**	68	64	61
pmed07	200	10	64	81	**20**	56	64	65
pmed08	200	20	55	51	**11**	421	83	108
pmed09	200	40	37	27	**21**	72	73	108
pmed10	200	67	20	12	**12**	57	49	60
pmed11	300	5	59	282	**19**	134	61	45
pmed12	300	10	51	248	**25**	117	75	142
pmed13	300	30	36	218	104	513	**97**	217
pmed14	300	60	26	127	**101**	429	117	390
pmed15	300	100	18	54	**55**	127	82	139
pmed16	400	5	47	875	**17**	194	63	141
pmed17	400	10	39	4652	113	298	**64**	224
pmed18	400	40	28	965	688	3601	**243**	445
pmed19	400	80	18	679	255	780	**173**	292
pmed20	400	133	13	165	44	194	**29**	238
pmed21	500	5	40	3001	**29**	65	69	68
pmed22	500	10	38	TL	381	770	**106**	775
pmed23	500	50	22	2729	456	1768	**310**	759
pmed24	500	100	15	1538	322	906	**130**	384
pmed25	500	167	11	574	70	378	**54**	687
pmed26	600	5	38	10,426	514	**105**	508	333
pmed27	600	10	32	TL	**41**	1140	86	138
pmed28	600	60	18	10,298	550	2906	**129**	883
pmed29	600	120	13	5195	**134**	1373	545	602
pmed30	600	200	9	1234	87	544	**60**	426
pmed31	700	5	30	TL	**34**	93	78	180
pmed32	700	10	29	TL	**1007**	4285	1514	1600
pmed33	700	70	15	TL	1086	3894	**397**	857
pmed34	700	140	11	8794	**761**	1026	**171**	818
pmed35	800	5	30	TL	2116	262	**129**	201
pmed36	800	10	27	TL	4243	4889	1875	**314**
pmed37	800	80	15	TL	1681	3922	**1595**	3251
pmed38	900	5	29	TL	**89**	1235	183	609
pmed39	900	10	23	TL	1932	**521**	1687	1932
pmed40	900	90	13	TL	**1419**	6397	1570	2341
Average				-	462	1095	**316**	498

Table 4.1: Performance comparison of (PCP) formulations. TL=7200s.

Instance				t[s]				
name	n=m	p	OPT	F1	F2	F3	F4	F5
pmed01	100	5	127	35	**8**	TL2	86	821
pmed02	100	10	98	29	**5**	TL2	70	TL2
pmed03	100	10	93	35	**4**	TL2	146	1677
pmed04	100	20	74	**6**	7	TL2	82	TL2
pmed05	100	33	48	**4**	5	TL2	50	TL2
pmed06	200	5	84	172	**44**	TL2	209	TL2
pmed07	200	10	64	1017	**25**	TL2	85	TL2
pmed08	200	20	55	1082	**32**	TL2	262	TL2
pmed09	200	40	37	568	**48**	TL2	285	TL2
pmed10	200	67	20	13	**20**	TL2	96	TL2
pmed11	300	5	59	TL2	**30**	TL2	99	TL2
pmed12	300	10	51	TL2	**72**	TL2	260	TL2
pmed13	300	30	36	TL2	**124**	TL2	474	TL2
pmed14	300	60	26	TL2	**101**	TL2	406	TL2
pmed15	300	100	18	TL2	**74**	TL2	215	TL2
pmed16	400	5	47	711	**27**	TL2	108	TL2
pmed17	400	10	39	TL2	**76**	TL2	588	TL2
pmed18	400	40	28	TL2	**163**	TL2	469	TL2
pmed19	400	80	18	TL2	**160**	TL2	589	TL2
pmed20	400	133	13	TL2	**109**	TL2	214	TL2
pmed21	500	5	40	TL2	**73**	TL2	122	TL2
pmed22	500	10	38	TL2	596	TL2	**574**	TL2
pmed23	500	50	22	TL2	**379**	TL2	1731	TL2
pmed24	500	100	15	TL2	**358**	TL2	1650	TL2
pmed25	500	167	11	TL2	**164**	TL2	208	TL2
pmed26	600	5	38	TL2	**132**	TL2	213	TL2
pmed27	600	10	32	TL2	974	TL2	**212**	TL2
pmed28	600	60	18	TL2	**1045**	TL2	TL2	TL2
pmed29	600	120	13	TL2	**177**	TL2	370	TL2
pmed30	600	200	9	TL2	283	TL2	**255**	TL2
pmed31	700	5	30	TL2	**64**	TL2	177	TL2
pmed32	700	10	29	TL2	**1013**	TL2	TL2	TL2
pmed33	700	70	15	TL2	**1443**	TL2	TL2	TL2
pmed34	700	140	11	TL2	**563**	TL2	998	TL2
pmed35	800	5	30	TL2	**82**	TL2	148	TL2
pmed36	800	10	27	TL2	**335**	TL2	905	TL2
pmed37	800	80	15	TL2	**1806**	TL2	TL2	TL2
pmed38	900	5	29	TL2	**86**	TL2	220	TL2
pmed39	900	10	23	TL2	**260**	TL2	696	TL2
pmed40	900	90	13	TL2	**1610**	TL2	TL2	TL2
	Average			-	314	-	-	-

Table 4.2: Performance comparison of (PCP) formulations without pre-solve. TL2=1800s.

4.4 Benders decomposition for a relaxation of (PCP)

Since decomposition algorithms are the state-of-the-art methods for solving the p-center problem and the formulation $(F2)$ has been shown to be efficient in the previous section, we focus on the Benders decomposition of $(F2)$. More details and references to the general Benders decomposition method are presented in the Section 2.2.1.

It is possible to perform two decompositions of Benders depending on which set of the two sets of variables from $(F2)$ are relaxed. In this Section 4.4 variables y are relaxed while in Section 4.5, we relax variables z.

The following formulation corresponds to $(F2)$ when the variables y are relaxed:

$$(F2_1)\begin{cases} \min & D^0 + \sum_{k\in\mathcal{K}} (D^k - D^{k-1}) z^k & & \\ \text{s.t.} & z^k + \sum_{j:d_{ij}<D^k} y_j \geq 1 & i \in \mathcal{N}, k \in \mathcal{K}_i & \\ & z^k \geq z^{k+1} & k \in \{1,2,\ldots,K-1\} & \\ & \sum_{j\in\mathcal{M}} y_j = p & & \\ & y_j \geq 0 & j \in \mathcal{M} & \\ & z^k \in \{0,1\} & k \in \mathcal{K} & \end{cases}$$

Consequently, the master problem associated with the variables z^k is:

$$(MP_1)\begin{cases} \min & D^0 + \sum_{k\in\mathcal{K}} (D^k - D^{k-1}) z^k & & \\ \text{s.t.} & z^k \geq z^{k+1} & k \in \{1,...,K-1\} & \\ & z \text{ satisfies } BD & & \\ & z^k \in \{0,1\} & k \in \mathcal{K} & \end{cases} \tag{4.30}$$

The sub-problem associated with a feasible solution $\bar{z}^k$ of (MP_1) is:

$$(SP_1)\begin{cases} \min & 0 & & \\ \text{s.t.} & \sum_{j\in\mathcal{M}} y_j = p & & \\ & \sum_{j:d_{ij}<D^k} y_j \geq (1-\bar{z}^k) & i \in \mathcal{N},\ k \in \mathcal{K}_i & \\ & y_j \geq 0 & j \in \mathcal{M} & \end{cases} \tag{4.31}$$

Consequently, the associated dual sub-problem is defined by:

$$(DSP_1)\begin{cases} \max & \sum_{i\in\mathcal{N}}\sum_{k\in\mathcal{K}_i} v_i^k(1-\bar{z}^k) + c*p & & (4.32)\\ \text{s.t.} & \sum_{i\in\mathcal{N}}\sum_{k\in\mathcal{K}_i:d_{ij}<D^k} v_i^k + c \le 0 & j\in\mathcal{M} & (4.33)\\ & v_i^k \ge 0 & i\in\mathcal{N},\ k\in\mathcal{K}_i & \\ & c\in\mathbb{R} & & \end{cases}$$

As (SP_1) is a feasibility problem, with an extreme ray $(\bar{v},\bar{c})$ of (DSP_1), we can only add to (MP_1) the following Benders feasibility cuts:

$$\sum_{i\in\mathcal{N}}\sum_{k\in\mathcal{K}_i} \bar{v}_i^k(1-z^k) + \bar{c}*p \le 0 \tag{4.34}$$

This decomposition provides a lower bound that corresponds to the solution of $(F2)$ (or $(F2_{init})$) in which the integrality of the y variables is removed. This lower bound is the a well-know tight lower bound of the p-center problem. It was introduced in Elloumi et al. (2004) and named LB^*.

4.4.1 Improved sub-problem formulation

The feasibility of (SP_1) depends on whether it is possible to find a solution in which the number of open sites is equal to p. It is then possible to remove the cardinality constraints on the number of sites and check afterwards if the number of sites required is equal to p or not. This leads to the following sub-problem:

$$(SP_1')\begin{cases} \min & \sum_{j\in\mathcal{M}} y_j & & \\ \text{s.t.} & \sum_{j:d_{ij}<D^k} y_j \ge (1-\bar{z}^k) & i\in\mathcal{N},\ k\in\mathcal{K}_i & (v_i^k)\\ & y_j \ge 0 & j\in\mathcal{M} & \end{cases}$$

The corresponding dual sub-problem is defined by:

$$(DSP_1')\begin{cases} \max & \sum_{i\in\mathcal{N}}\sum_{k\in\mathcal{K}_i} v_i^k(1-\bar{z}^k) & & (4.35)\\ \text{s.t.} & \sum_{i\in\mathcal{N}}\sum_{k\in\mathcal{K}_i:d_{ij}<D^k} v_i^k \le 1 & j\in\mathcal{M} & (4.36)\\ & v_i^k \ge 0 & i\in\mathcal{N},\ k\in\mathcal{K}_i & \end{cases}$$

The advantage of this reformulation is that the variable c is not required anymore. Consequently, the sub-problem is feasible for any solution of the master problem. Nevertheless, the objective

value of the primal or dual sub-problem must be equal to p to be a feasible solution of the original problem. If this condition is not satisfied, we can then add the following cut to the master problem:

$$\sum_{i \in \mathcal{N}} \sum_{k \in \mathcal{K}_i} \bar{v}_i^k (1 - z^k) \leq p \tag{4.37}$$

This is also known as a normalization of the feasibility cuts, when a constraint of $c = 1$ is added to $(DSP1)$. We can solve $(DSP1')$ more quickly thanks to the following proposition.

Proposition 4.4.1. *There exists an optimal solution $\bar{v}$ of the sub-problem $(DSP1')$ with at most one $k \in \mathcal{K}_i$ such that $\bar{v}_i^k > 0$ for all $i \in \mathcal{N}$.*

Proof:

Given $i \in \mathcal{N}$ and $k \in \mathcal{K}$, if $(1 - \bar{z}^k) \leq 0$ and $k \in \mathcal{K}_i$, then there exists necessarily an optimal solution of $(DSP1')$ in which $\bar{v}_i^k = 0$. Similarly, in an optimal solution solution $v_i^k > 0$ implies $(1 - \bar{z}^k) > 0$ which is possible only if $\bar{z}^k = 0$.

Let us assume that there exists an optimal solution $\bar{v}$, such that $\bar{v}_i^{k_1} > 0$ and $\bar{v}_i^{k_2} > 0$, with $k_1 < k_2$. Then, there exists j_1 and j_2, such that exists $d_{ij_1} = D^{k_1}$, $d_{ij_2} = D^{k_2}$ and $d_{ij_1} < d_{ij_2}$.

Note that if variable $v_i^{k_1}$ appears in a given constraint of $(DSP1')$, so does $v_i^{k_2}$ with the same coefficients. Since they have the same coefficient in the objective, it is possible to build a solution v^* equal to $\bar{v}$ except that $v_i^{*k_1} = \bar{v}_i^{k_1} + \bar{v}_i^{k_2}$ and $v_i^{*k_2} = 0$. Thus, v^* satisfies Constraints (4.36) and it has the same optimal objective value. This process can be repeated until for each index i, at most one variable v_i^k is greater than 0. □

Proposition 4.4.1 allows us to obtain an optimal solution of $(DSP1')$ by solving a sub-problem with fewer variables where for each client $i \in \mathcal{N}$ a single variable $v_i^{\underline{k}}$ is considered, where $\underline{k}$ is the smallest distance index $k \in \mathcal{K}$ such that $\bar{z}^k = 0$:

$$(DSP_1^*) \begin{cases} \max & \displaystyle\sum_{i \in \mathcal{N}} v_{i\underline{k}} & & (4.38) \\ \text{s.t.} & \displaystyle\sum_{i \in \mathcal{N}: d_{ij} < D^{\underline{k}}} v_{i\underline{k}} \leq 1 & j \in \mathcal{M} & (4.39) \\ & v_{i\underline{k}} \geq 0 & i \in \mathcal{N} & \end{cases}$$

with $\underline{k} = \min\{k \in \mathcal{K} : \bar{z}^k = 0\}$ for all $i \in \mathcal{N}$. Note (DSP_1^*) is the linear relaxation of the maximal stable set problem in the graph where the nodes are the clients and an edge links two clients that can be covered by the same site at a distance lower than $D^{\underline{k}}$.

Consequently, the Benders cut associated with the optimal solution of $(DSP1')$ $\bar{v}_i^k$ can be rewritten as follows:

$$\sum_{i \in \mathcal{N}} \bar{v}_i^k (1 - z^k) \leq p \tag{4.40}$$

4.4.2 Efficient master problem resolution algorithm

Usually, at each step of the Benders algorithm, the master problem and a sub-problem need to be solved. (MP_1) is an easy problem to solve since it is a minimization problem with positive coefficients in the objective, therefore the variables take the value 0 unless there is a constraint that does not allow it. Thus, the optimal solution is the smallest distance allowed by the cut added in the last iteration. Note that the cut (4.40) can be written as follows:

$$\sum_{i \in \mathcal{N}} \bar{v}_i^k z^k \geq \sum_{i \in \mathcal{N}} \bar{v}_i^k - p$$

Consequently, given Constraints (4.30), the optimal solution of $(DPS1')$ can be computed by successively setting the value of $z^k = 1$ from the smallest index k that appear in the cut until the inequality is satisfied. Moreover, this decomposition can be implemented as a linear search of the distance D^k until a feasible solution is found. Indeed distance D^0 is initially returned and at each iteration k, distance D^k is obtained until the optimal radius is reached. In the next section we will show the relation of this to a classical solving algorithm of using a binary search.

4.4.3 Binary search algorithm

As mentioned in the literature review, a classical method to solve the p-center problem is using the relation with the *Set Cover Problem* (SCP) (Toregas et al. (1971)). The (SCP) associated with a distance D^k is formulated as follows:

$$\left(SCP_{(D^k)}\right) \begin{cases} \min & \displaystyle\sum_{j \in \mathcal{M}} y_j & \\ \text{s.t.} & \displaystyle\sum_{j : d_{ij} \leq D^k} y_j \geq 1 & i \in \mathcal{N} \\ & y_j \in \{0,1\} & j \in \mathcal{M} \end{cases}$$

The radius of (PCP) is at most D^k, if and only if the solution value of $(SCP_{(D^k)})$ is at most p. The smallest distance D^k that needs at most p centers is the optimal solution of the corresponding (PCP). To find this minimum distance D^k, a binary search is used so that at most $log_2(K+1)$ set cover problems need to be solved. Elloumi et al. (2004) proposes a two-step solution, in which

first the variables y_j are relaxed and together with heuristics to obtain good lower and upper bounds, and thus reduce the number of distances for solution in integer variables in the second stage.

Algorithm 5: General binary search algorithm

input : (PCP) instance data ($\mathcal{N}$, $\mathcal{M}$, p, $D^1, ..., D^K$)

output: An optimal solution with centers y^* of cardinality p and radius θ^*

$l \leftarrow 1$, $r \leftarrow K$

while $l < r$ **do**

  $h \leftarrow \lfloor (l + r)/2 \rfloor$

  $\theta \leftarrow D^h$

  $y \leftarrow SCP(D^h)$ // Solving the corresponding set cover problem

  if $|y| > p$ **then**

    $l \leftarrow h + 1$

  else

    $r \leftarrow h$

    $(\theta^*, y^*) \leftarrow (\theta, y)$

return (θ^*, y^*)

We observe experimentally that this first Benders decomposition on $(F2)$ is slower than this general binary search method with the set cover sub-problems. The performance of this implementation may be improved in the future by using specialized techniques in the solution of maximal stable set problems, for example. In the following section we detail a second Benders decomposition in which variables z^k are relaxed in the sub-problems instead of variables y_j.

4.5 Benders decomposition for an exact relaxation of (PCP)

In this section we present the second Benders decomposition on formulation $(F2)$. Gaar and Sinnl (2022) consider a Benders decomposition of $(F1)$ which leads to formulation $(F5)$. We show that $(F5)$ is also related to this decomposition. We first relax variables z^k in $(F2)$.

$$(F2_2)\begin{cases} \min & D^0 + \sum_{k\in\mathcal{K}}(D^k - D^{k-1})z^k & & \\ \text{s.t.} & z^k + \sum_{j:d_{ij}<D^k} y_j \geq 1 & i\in\mathcal{N}, k\in\mathcal{K}_i \\ & z^k \geq z^{k+1} & k\in\{1,2,\dots,K-1\} \\ & \sum_{j\in\mathcal{M}} y_j = p & \\ & y_j \in \{0,1\} & j\in\mathcal{M} \\ & z^k \geq 0 & k\in\mathcal{K} \end{cases}$$

The master problem associated with variables y_j is:

$$(MP_2)\begin{cases} \min & \theta & & \\ \text{s.t.} & \sum_{j\in\mathcal{M}} y_j = p & \\ & y_j \in \{0,1\} & j\in\mathcal{M} \end{cases}$$

Given a solution $\bar{y}$ of (MP_2), the sub-problem is defined by:

$$(SP_2)\begin{cases} \min & D^0 + \sum_{k\in\mathcal{K}}(D^k - D^{k-1})z^k & & & & \\ \text{s.t.} & z^k - z^{k+1} \geq 0 & k\in\{1,...,K-1\} & (b_k) & (4.41) \\ & z^k \geq 1 - \sum_{j:d_{ij}<D^k} \bar{y}_j & i\in\mathcal{N}, k\in\mathcal{K}_i & (v_i^k) & (4.42) \\ & z^k \geq 0 & k\in\mathcal{K} & & \end{cases}$$

and its associated dual problem is:

$$(DSP_2)\begin{cases} \max & D^0 + \sum_{i\in N}\sum_{k\in\mathcal{K}_i} v_i^k(1 - \sum_{j:d_{ij}<D^k} \bar{y}_j) & & & (4.43) \\ \text{s.t.} & b_1 + \sum_{i\in N:1\in\mathcal{K}_i} v_i^1 \leq D^1 - D^0 & k=1 & & (4.44) \\ & b_k - b_{k-1} + \sum_{i\in N:k\in\mathcal{K}_i} v_i^k \leq D^k - D^{k-1} & k=2,...,K-1 & & (4.45) \\ & -b_{K-1} + \sum_{i\in N:K\in\mathcal{K}_i} v_i^K \leq D^K - D^{K-1} & k=K & & (4.46) \\ & b_k \geq 0 & k\in\{1,...,K-1\} & & \\ & v_i^k \geq 0 & i\in\mathcal{N}, k\in\mathcal{K}_i & & \end{cases}$$

The optimality cut associated with an optimal solution $\bar{v}_i^k$ of (DSP_2) is:

$$\theta \geq D_0 + \sum_{i \in N} \sum_{k \in \mathcal{K}_i} \bar{v}_i^k (1 - \sum_{j : d_{ij} < D^k} y_j) \tag{4.47}$$

The resolution of this Benders decomposition gives us the optimal solution of $(F2)$, since the integrality of variables z^k are ensured by that of variables y_j.

4.5.1 Efficient sub-problem resolution algorithm

We show that the sub-problem of this decomposition can be solved quickly.

Lemma 4.5.1. *Given a solution $\bar{y}$ of the master problem (MP_2) or of its LP-relaxation, the following solution $\bar{z}$ is feasible for (SP_2):*

$$\bar{z}^k = \max \left(0, \max_{i \in \mathcal{N}} (1 - \sum_{j : d_{ij} < D^k} \bar{y}_j)\right) \qquad k \in \mathcal{K} \tag{4.48}$$

Proof. All $\mathcal{K}_i$ indices are included in $\mathcal{K}$ for at least one client, and it holds that

$$(1 - \sum_{j : d_{ij} < D^k} \bar{y}_j) \geq (1 - \sum_{j : d_{ij} < D^{k+1}} \bar{y}_j) \qquad k \in \{1, ..., K-1\}$$

Therefore, the solution $\bar{z}$ satisfies Constraints (4.41) and (4.42) of $(SP1)$. □

Note that in this solution, the value of variables $\bar{z}^k$ are not increasing when k increases. In order to obtain a complementary dual solution, we will see that it is relevant to identify the last strictly positive term of this sequence.

Definition 4.5.1. *Given a solution $\bar{y}$ of the master problem (MP_2) or of its LP-relaxation, let $\tilde{k}$ be the following index:*

$$\tilde{k} = \max \left\{ k \in \mathcal{K} : \left(\max_{i \in \mathcal{N}} (1 - \sum_{j : d_{ij} < D^k} \bar{y}_j) \right) > 0 \right\} \tag{4.49}$$

Note that, if $\bar{y}$ is binary, then distance $D^{\tilde{k}}$ corresponds to the radius of the feasible solution $\bar{y}$. For each index $k \leq \tilde{k}$, we are also interested in identifying a client that limits the reduction of $\bar{z}^k$.

Definition 4.5.2. *Given a solution $\bar{y}$ of the master problem (MP_2) or of its LP-relaxation, and the associated index $\tilde{k}$ for each $k \in \{1, ..., \tilde{k}\}$, we define $i^*_{(k)}$ as:*

$$i^*_{(k)} \in \arg\max_{i \in \mathcal{N}} (1 - \sum_{j: d_{ij} < D^k} \bar{y}_j) \tag{4.50}$$

Client $i^*_{(k)}$ is a client that will be covered at least at the distance D^k given solution $\bar{y}$. Considering these definitions, we can express the optimal value of solution $\bar{z}$.

Lemma 4.5.2. *Given a solution $\bar{y}$ of the master problem (MP_2) or of its LP-relaxation, and a corresponding client $i^*_{(k)}$ for $k \in \{1, ..., \tilde{k}\}$, the objective value of solution $\bar{z}$ is:*

$$\theta^{SP_2}_{(\bar{y})} = D^{\tilde{k}} - \sum_{k=1}^{\tilde{k}} (D^k - D^{k-1}) \sum_{j: d_{i^*_{(k)} j} < D^k} \bar{y}_j \tag{4.51}$$

Proof. For $k > \tilde{k}$, we have that $\bar{z}^k = 0$ and $\bar{z}^k > 0$ for $k \leq \tilde{k}$. Consequently, the objective value of $\bar{z}$ can be written as:

$$\begin{aligned}
\theta^{SP_2}_{(\bar{y})} &= D^0 + \sum_{k \in \mathcal{K}} (D^k - D^{k-1}) \Big(\max \Big(0, \max_{i \in \mathcal{N}} (1 - \sum_{j: d_{ij} < D^k} \bar{y}_j) \Big) \\
&= D^0 + \sum_{k=1}^{\tilde{k}} (D^k - D^{k-1}) \Big(\max_{i \in \mathcal{N}} (1 - \sum_{j: d_{ij} < D^k} \bar{y}_j) \Big) \\
&= D^0 + \sum_{k=1}^{\tilde{k}} (D^k - D^{k-1}) (1 - \sum_{j: d_{i^*_{(k)} j} < D^k} \bar{y}_j) \\
&= D^{\tilde{k}} - \sum_{k=1}^{\tilde{k}} (D^k - D^{k-1}) \sum_{j: d_{i^*_{(k)} j} < D^k} \bar{y}_j
\end{aligned}$$

□

The following proposition allows us to define an optimal solution of $(DSP2)$.

Proposition 4.5.1. *Given a solution $\bar{y}$ of the master problem (MP_2) or of its LP-relaxation, and its corresponding indices $i^*_{(k)}$ for $k \in \{1, ..., \tilde{k}\}$, the solution $\bar{z}$ is optimal for (SP_2) and the following solution $(\bar{v}, \bar{b})$ is optimal for (DSP_2) for $k \in \mathcal{K}$:*

$$\bar{b}_k = 0 \qquad\qquad \bar{v}^k_i = \begin{cases} D^k - D^{k-1} & \text{if } k \leq \tilde{k} \text{ and } i = i^*_{(k)} \\ 0 & \text{otherwise} \end{cases}$$

Proof. By definition of $\tilde{k}$, client $i^*_{(k)}$ exists for each $k \leq \tilde{k}$. It is thus easy to check that the solution $(\bar{v}, \bar{b})$ is feasible. The objective value of this solution is:

$$\begin{aligned}
D^0 + \sum_{i\in N}\sum_{k\in\mathcal{K}_i} \bar{v}_i^k(1 - \sum_{j:d_{ij}<D^k} \bar{y}_j) =& D^0 + \sum_{i\in N}\sum_{k=1}^{\tilde{k}} \bar{v}_i^k(1 - \sum_{j:d_{ij}<D^k} \bar{y}_j) \\
=& D^0 + \sum_{k=1}^{\tilde{k}} (D^k - D^{k-1})(1 - \sum_{j:d_{i^*_{(k)}j}<D^k} \bar{y}_j) \\
=& D^{\tilde{k}} - \sum_{k=1}^{\tilde{k}} (D^k - D^{k-1}) \sum_{j:d_{i^*_{(k)}j}<D^k} \bar{y}_j \\
=& \theta^{SP_2}_{(\bar{y})}
\end{aligned}$$

Hence, this dual solution is feasible and it has the same objective value as the primal solution $\bar{z}$. Therefore, the solutions $\bar{z}$ and $(\bar{v}, \bar{b})$ are optimal for (SP_2) and (DSP_2), respectively. □

Corollary 1. The Benders cuts (4.47) can be written as follows:

$$\theta \geq D^{\tilde{k}} - \sum_{k=1}^{\tilde{k}} (D^k - D^{k-1})(\sum_{j:d_{i^*_{(k)}j}<D^k} y_j) \tag{4.52}$$

where $\tilde{k}$ and $i^*_{(k)}$ are defined by Definitions 4.5.1 and 4.5.2. Note that the coefficients of these cuts are integers even if the master problem is solved in continuous variables.

4.5.2 Representation of the cuts

Note that for a given $k \leq \tilde{k}$ diferent clients can be chosen as $i^*_{(k)}$. Moreover, if i can be chosen as $i^*_{(k)}$, it can also be chosen as $i^*_{(k-1)}$, for all $k \in \{2, ..., \tilde{k}\}$, if $\bar{y}$ is integer. This allows us to define the following corollary.

Corollary 2. In the case where $i^*_{(k)} = i^*_{(\tilde{k})}$ for each $k \leq \tilde{k}$, the cuts (4.52) can be rewritten as follows.

$$\begin{aligned}
\theta \geq& D^{\tilde{k}} - \sum_{k=1}^{\tilde{k}} (D^{\tilde{k}} - D^k)(\sum_{j:d_{i^*_{(\tilde{k})}j}=D^k} y_j) \\
\theta \geq& D^{\tilde{k}} - \sum_{j:d_{i^*_{(\tilde{k})}j}<D^{\tilde{k}}} (D^{\tilde{k}} - d_{i^*_{(\tilde{k})}j}) y_j
\end{aligned} \tag{4.53}$$

The cuts (4.53) indicate that the radius is at least $D^{\tilde{k}}$ unless a site j such that $d_{i^*_{(\tilde{k})}j} < D^{\tilde{k}}$ is opened, since client $i^*_{(\tilde{k})}$ is the one inducing this current maximal allocation distance. In that case, the allocation distance of client $i^*_{(\tilde{k})}$ could be reduced to $d_{i^*_{(\tilde{k})}j}$.

Moreover, the cuts (4.53) are included in the formulation $(F5)$, introduced in Gaar and Sinnl (2022) for the (PCP). To see this relation directly, we write as $(F5')$ the (PCP) formulation obtained with cuts (4.53).

$$(F5')\begin{cases} \min & \theta & & \\ \text{s.t.} & \theta \geq D^k - \sum_{j:d_{ij}<D^k} (D^k - d_{ij})y_j & i \in \mathcal{N}, k \in \mathcal{K}_i & (4.54) \\ & \sum_{j\in\mathcal{M}} y_j = p & & \\ & y_j \in \{0,1\} & j \in \mathcal{M} & \end{cases}$$

It is important to note that formulation $(F5')$ has the same linear relaxation value as $(F1)$. Moreover, if in our Benders decomposition we relax the master problem and separate only the cuts (4.53), this same lower bound from $(F1)$ would be obtained. However, if the cuts (4.52) are separated, we obtain the linear relaxation bound of $(F2)$, which is better than $(F1)$ (Elloumi et al. (2004); Ales and Elloumi (2018)).

To illustrate the differences between the approaches to generating the cuts, let us consider the following instance of 6 clients and sites:

$$d_{ij} = \begin{pmatrix} 526 & 202 & 399 & 699 & 383 & 629 \\ 156 & 397 & 743 & 748 & 292 & 652 \\ 741 & 465 & 55 & 427 & 446 & 396 \\ 966 & 913 & 656 & 280 & 681 & 322 \\ 350 & 213 & 688 & 915 & 448 & 828 \\ 1003 & 761 & 264 & 361 & 678 & 404 \end{pmatrix} \quad (4.55)$$

All distances are different, therefore we have the following 36 distinct distance values:

$$\begin{aligned} \{D^k\} = \{&55, 156, 202, 213, 264, 280, 292, 322, 350, 361, 383, 396, \\ &397, 399, 404, 427, 446, 448, 465, 526, 629, 652, 656, 678, \\ &681, 688, 699, 741, 743, 748, 761, 828, 913, 915, 966, 1003\} \end{aligned}$$

For $p = 3$ this instance has an optimal value of $D^{12} = 397$, a linear relaxation value of $D^5 = 280$ and 319.666 (close to D^7) from formulations $(F1)$ and $(F2)$, respectively.

If we use our Benders decomposition with cuts (4.52), we obtain the following 7 cuts:

1. $i^*_{(k\in\{1,\dots,2\})} = 1,\ i^*_{(k\in\{3,\dots,\tilde{k}=22\})} = 4 : \quad \theta + 376y_4 + 334y_6 \geq 656$
2. $i^*_{(k\in\{1,\dots,2\})} = 1,\ i^*_{(k\in\{3,\dots,\tilde{k}=12\})} = 2 : \quad \theta + 195y_1 + 105y_5 \geq 397$
3. $i^*_{(k\in\{1,\dots,\tilde{k}=19\})} = 1 : \quad \theta + 324y_2 + 127y_3 + 143y_5 \geq 526$
4. $i^*_{(k\in\{1,\dots,2\})} = 1,\ i^*_{(k\in\{3,\dots,\tilde{k}=15\})} = 3 : \quad \theta + 225y_3 + 31y_6 \geq 427$
5. $i^*_{(k\in\{1,\dots,10\})} = 1,\ i^*_{(k\in\{11,\dots,\tilde{k}=17\})} = 5 : \quad \theta + 65y_1 + 246y_2 \geq 448$
6. $i^*_{(k\in\{1,\dots,10\})} = 1,\ i^*_{(k=11)} = 3,\ i^*_{(k\in\{12,\dots,\tilde{k}=14\})} = 6 : \quad \theta + 181y_2 + 21y_3 + 8y_4 \geq 404$
7. $i^*_{(k\in\{1,\dots,2\})} = 1,\ i^*_{(k\in\{3,\dots,11\})} = 3,\ i^*_{(k\in\{12,\dots,\tilde{k}=14\})} = 6 : \quad \theta + 202y_3 + 8y_4 \geq 404$

If we now use cuts (4.53), we obtain the following 6 cuts:

1. $i^*_{(k\in\{1,\dots,\tilde{k}=22\})} = 4 : \quad \theta + 376y_4 + 334y_6 \geq 656$
2. $i^*_{(k\in\{1,\dots,\tilde{k}=12\})} = 2 : \quad \theta + 241y_1 + 105y_5 \geq 397$
3. $i^*_{(k\in\{1,\dots,\tilde{k}=19\})} = 1 : \quad \theta + 324y_2 + 127y_3 + 143y_5 \geq 526$
4. $i^*_{(k\in\{1,\dots,\tilde{k}=15\})} = 3 : \quad \theta + 372y_3 + 31y_6 \geq 427$
5. $i^*_{(k\in\{1,\dots,\tilde{k}=17\})} = 5 : \quad \theta + 98y_1 + 235y_2 \geq 448$
6. $i^*_{(k\in\{1,\dots,\tilde{k}=14\})} = 6 : \quad \theta + 140y_3 + 43y_4 \geq 404$

We can note the difference in the coefficients in most the constraints. For example, at iteration 5 $\bar{y} = (0, 0, 1, 1, 1, 0)$, i.e., the sites 3, 4 and 5 are opened. For this solution the radius is $D^{16} = 448$. For the first approach, the cut (4.52) considers $i^*_{(k)} = 1$ for $1 \leq k \leq 10$ and $i^*_{(k)} = 5$ for $11 \leq k \leq 17$. Meanwhile, for the cut (4.53) we consider $i^*_{(k)} = i^*_{(\tilde{k}=17)} = 5$ for all $k \leq \tilde{k}$, which leads to different linear relaxation values for the master problem.

In this example, due to the choices of $i^*_{(k)}$ for all $k \leq \tilde{k}$ at each iteration, the Benders decomposition based on cuts (4.52) leads to more iterations that the one based on cuts (4.53). However, since cuts (4.53) are included in (4.52), there necessarily exist choices of $i^*_{(k)}$ that enable to obtain the same number of iterations (and potentially fewer). Moreover, (4.52) are included in Constraints (4.54) of formulation $(F5')$. However, note that the cuts (4.52) depend on a feasible solution $\bar{y}$, which is not the case of the constraints in $(F5')$.

Now, if we relax the integrality of y in the master problem, we obtain 15 cuts (4.52) and 8 cuts (4.53). For example, in the corresponding second iteration both consider the fractional solution $\bar{y} = (0.1617, 1.0, 0.0, 1.0, 0.0, 0.8383)$, i.e., to open (fractionally) sites 1, 2, 4, and 6. The first cut (4.52) is:

$$\theta + y_1 + 224y_3 + y_5 + 30y_6 \geq 427$$

With a sub-problem solution value of 401.6886, this cut considers:

$$i^*_{(k)} = \begin{cases} 1 & 1 \leq k \leq 2 \\ 3 & 3 \leq k \leq 11 \\ 2 & k = 12 \\ 3 & 13 \leq k \leq 15 \end{cases}$$

While if only $i^*_{(\tilde{k}=15)} = 3$ is considered, the following cut (4.53) is obtained

$$\theta + 372y_3 + 31y6 \geq 427$$

with a sub-problem solution value of 401.0120. Although more cuts are needed for the first approach, an objective value of 319.666 is obtained while in the second one we obtain a value of 280. These values correspond to the relaxation bound of $(F2)$ and $(F1)$, respectively. The following section shows how to strengthen these different cuts.

4.5.3 Lifting procedure

In the PCP, if a lower bound LB of the optimal radius is known, all distances that are lower than LB can be replaced by LB. This is presented and exploited in the lifting procedure proposed by Gaar and Sinnl (2022). Consequently, formulation $(F5')$ can be strengthened as follows.

$$(\overline{F5'}) \begin{cases} \min & \theta & & \\ \text{s.t.} & \theta \geq D^k - \sum_{j:d_{ij}<D^k} (D^k - \max\{d_{ij},\ LB\})y_j & i \in \mathcal{N}, k \in \mathcal{K} : D^k > LB & (4.56) \\ & \theta \geq LB & & (4.57) \\ & \sum_{j \in \mathcal{M}} y_j = p & & \\ & y_j \in \{0,1\} & j \in \mathcal{M} & \end{cases}$$

Gaar and Sinnl (2022) update the lower bound LB with the value of the master problem solution at each iteration of the cut separation step in a *branch-and-cut* approach. We can also use this procedure to strengthen our Benders cuts (4.52) as follows:

$$\theta \geq D^{\tilde{k}} - \sum_{k=1}^{\tilde{k}} (\max\{D^k,\ LB\} - \max\{D^{k-1},\ LB\})\Big(\sum_{j:d_{i^*_{(k)}j}<\max\{D^k,\ LB\}} y_j\Big) \qquad (4.58)$$

Similarly, the lifted version of cuts (4.53) is:

$$\theta \geq D^{\tilde{k}} - \sum_{j:d_{i^*_{(\tilde{k})}j}<D^{\tilde{k}}} (D^{\tilde{k}} - \max\{d_{i^*_{(\tilde{k})}j},\ LB\})\bar{y}_j \tag{4.59}$$

In order to illustrate the impact of the lifting on the Benders decomposition, we now consider them in a Benders decomposition with an integer y on instance (4.55) with strengthened cuts (4.58) we obtain:

1. $i^*_{(k\in\{1,\dots,2\})} = 1,\ i^*_{(k\in\{3,\dots,\tilde{k}=22\})} = 4,\quad LB = 0:\quad \theta + 376y(3) + 334y(5) \geq 656$
2. $i^*_{(k\in\{1,\dots,2\})} = 1,\ i^*_{(k\in\{3,\dots,\tilde{k}=12\})} = 2,\quad LB = D^0 = 55:\quad \theta + 195y(0) + 105y(4) \geq 397$
3. $i^*_{(k\in\{2,\dots,\tilde{k}=19\})} = 1,\quad LB = D^2 = 202:\quad \theta + 324y(1) + 127y(2) + 143y(4) \geq 526$
4. $i^*_{(k\in\{5,\dots,\tilde{k}=15\})} = 3,\quad LB = D^5 = 280:\quad \theta + 147y(2) + 31y(5) \geq 427$
5. $i^*_{(k\in\{6,\dots,10\})} = 1,\ i^*_{(k\in\{11,\dots,\tilde{k}=17\})} = 5,\quad LB = D^6 = 292:\quad \theta + 65y(0) + 156y(1) \geq 448$
6. $i^*_{(k\in\{11,\dots,\tilde{k}=14\})} = 6,\quad LB = D^{11} = 396:\quad \theta + 8y(2) + 8y(3) \geq 404$

with the strengthened cuts (4.59) are:

1. $i^*_{(k\in\{1,\dots,\tilde{k}=22\})} = 4,\quad LB = 0:\quad \theta + 376y(3) + 334y(5) \geq 656$
2. $i^*_{(k\in\{1,\dots,\tilde{k}=12\})} = 2,\quad LB = D^0 = 55:\quad \theta + 241y(0) + 105y(4) \geq 397$
3. $i^*_{(k\in\{1,\dots,\tilde{k}=19\})} = 1,\quad LB = D^1 = 156:\quad \theta + 324y(1) + 127y(2) + 143y(4) \geq 526$
4. $i^*_{(k\in\{5,\dots,\tilde{k}=15\})} = 3,\quad LB = D^5 = 280:\quad \theta + 147y(2) + 31y(5) \geq 427$
5. $i^*_{(k\in\{6,\dots,\tilde{k}=17\})} = 5,\quad LB = D^6 = 292:\quad \theta + 98y(0) + 156y(1) \geq 448$
6. $i^*_{(k\in\{11,\dots,\tilde{k}=14\})} = 6,\quad LB = D^{11} = 396:\quad \theta + 8y(2) + 8y(3) \geq 404$

We see how the LB value increases to 396 in both cases. This value corresponds to LB^* introduced in Elloumi et al. (2004) and also presented in Gaar and Sinnl (2022) as the best known lower bound for the (PCP). Note that as LB increases, the coefficients of the variables y_j in the cuts decrease. The strengthening of cuts (4.52) enables to reduce by one the number of iterations.

This lifting procedure can also be used for fractional solutions, the values of 361 and 350 are obtained as relaxations bound of the master problem for both approaches instead of 319,666 and 280, respectively. In general, this procedure considerably reduces the number of iterations and thus the total computation time.

4.5.4 Separation algorithms

In the following, we present our algorithms to obtain the different Benders cuts introduced previously. Algorithm 6 presents our separation algorithm for the strengthened cuts (4.58) and Algorithm 7 for the strengthened cuts (4.59). Moreover, Algorithm 8 presents how to separate one strengthened Benders cut (4.59) for each client $i \in N$.

In Algorithm 6, for each value of k from the distance index k^{LB} of the lower bound $LB = D^{(k^{LB})}$, the value $\beta_k = \max_{i \in \mathcal{N}} (1 - \sum_{j:d_{ij}<D^k} \bar{y}_j)$ and an associated client $i^*_{(k)}$ are calculated (Steps 3 to 8). Then, we sequentially compute the sub-problem objective value $\theta^{(SP)}$ and construct the corresponding Benders cut $\theta \geq w(y)$ (Steps 9 to 17). If the cut is violated by the solution of the master problem (i.e., $\theta^{(SP)} > LB$), then the cut is added to the (MP) (Step 18 to 19).

Algorithm 6: Separation algorithm for a lifted single cut (4.58)

input :
- Instance data $\big(\mathcal{N}, \mathcal{M}, \mathcal{K}$, distances $D^0, ..., D^K$ and d_{ij} for each $i \in \{1...N\}$, $j \in \{1...M\}\big)$
- Current (MP) solution $\bar{y}$ and a lower bound (LB, k^{LB})

output:
- $\tilde{k}$ index and the sub-problem objective value $\theta^{(SP)}$

1 $\theta^{(SP)} \leftarrow LB$
2 **for** $k \in \{k^{LB} + 1, ..., \mathcal{K}\}$ **do**
3 $\quad \beta_k \leftarrow 0$ // Compute $i^*_{(k)}$ and the $\beta_k = \max_{i \in \mathcal{N}} (1 - \sum_{j:d_{ij}<D^k} \bar{y}_j)$
4 $\quad$ **for** $i \in \mathcal{N}$ **do**
5 $\quad\quad \beta_{ik} \leftarrow 1 - \sum_{j:d_{ij}<D^k} \bar{y}_j$
6 $\quad\quad$ **if** $\beta_{ik} > \beta_k$ **then**
7 $\quad\quad\quad \beta_k \leftarrow \beta_{ik}$
8 $\quad\quad\quad i^*_{(k)} \leftarrow i$
9 $\quad$ **if** $\beta_k > 0$ **then**
10 $\quad\quad \tilde{k} \leftarrow k$ // Compute $\theta^{(SP)}$ and generate the cut $\theta \geq w(y)$
11 $\quad\quad v_i^k \leftarrow \max\{D^k, LB\} - \max\{D^{k-1}, LB\}$
12 $\quad\quad \theta^{(SP)} \leftarrow \theta^{(SP)} + \beta_k v_i^k$
13 $\quad\quad$ **for** $j \in \mathcal{M}$ **do**
14 $\quad\quad\quad$ **if** $d_{i^*_{(k)}j} < \max\{D^k, LB\}$ **then**
15 $\quad\quad\quad\quad w(y) \leftarrow w(y) + v_i^k y_j$
16 $\quad\quad$ **else**
17 $\quad\quad\quad$ break
18 **if** $\theta^{(SP)} > LB$ **then**
19 $\quad$ Add $\theta \geq D^{\tilde{k}} - w(y)$ to (MP)
20 **return** $(\tilde{k}, \theta^{(SP)})$

In Algorithm 7, for each client $i \in \mathcal{N}$, we compute the value $\beta_{ik} = \max\left(0,\ (1 - \sum_{j:d_{ij}<D^k} \bar{y}_j)\right)$ for $k \in \mathcal{K}$ calculating the client allocation distance θ_i (Step 3 to 10). We then update the sub-problem objective value $\theta^{(SP)}$ and the client $i^*_{\tilde{k}}$ (Steps 11 to 13). If the cut is violated by the solution of the master problem, then the Benders cut associated to $i^*_{\tilde{k}}$ is added to the (MP) (Step 14 to 15).

Algorithm 7: Separation algorithm for a lifted single cut (4.59)

input :

- Instance data $\Big(\mathcal{N}$, $\mathcal{M}$, $\mathcal{K}$, distances $D^0, ..., D^K$ and d_{ij} for each $i \in \{1...N\}$, $j \in \{1...M\}\Big)$
- Current (MP) solution $\bar{y}$ and a lower bound (LB, k^{LB})

output:

- $\tilde{k}$ index, client $i^*_{(\tilde{k})}$ index and the sub-problem objective value $\theta^{(SP)}$

1 $\theta^{(SP)} \leftarrow LB$
2 **for** $i \in \mathcal{N}$ **do**
3 $\quad\theta_i \leftarrow LB$
4 $\quad\beta_{ik} \leftarrow 1$
5 $\quad k = k^{LB}$
6 $\quad$**while** $\beta_{ik} > 0$ **do**
7 $\quad\quad k \leftarrow k + 1$
8 $\quad\quad\beta_{ik} \leftarrow \beta_{ik} - \sum_{j:d_{ij}=D^{k-1}} \bar{y}_j$
9 $\quad\quad\theta_i \leftarrow \theta_i + (D^k - D^{k-1})\max(0,\ \beta_{ik})$
10 $\quad\quad\tilde{k} \leftarrow k$
11 $\quad$**if** $\theta_i > \theta^{(SP)}$ **then**
12 $\quad\quad\theta^{(SP)} \leftarrow \theta_i$
13 $\quad\quad i^*_{\tilde{k}} \leftarrow i$
14 **if** $\theta^{(SP)} > LB$ **then**
15 $\quad$Add $\theta \geq D^{\tilde{k}} - \sum_{j:d_{i^*_{(\tilde{k})}j}<D^{\tilde{k}}} (D^{\tilde{k}} - \max\{d_{i^*_{(\tilde{k})}j},\ LB\})y_j$ to (MP)
16 **return** $(\tilde{k}, \theta^{(SP)})$

Algorithm 8, is similar to Algorithm 7, we compute the value $\beta_{ik} = \max\left(0,\ (1 - \sum_{j:d_{ij}<D^k} \bar{y}_j)\right)$ for each client $i \in \mathcal{N}$ and $k \in \mathcal{K}$ (Steps 3 to 12). Nevertheless, we evaluate directly if the client allocation distance θ_i is greater than LB. If it is the case, the corresponding Benders cut for the client i is added to the master problem and the sub-problem objective value $\theta^{(SP)}$ is updated (Steps 14 to 16).

Algorithm 8: Separation algorithm with multi-cut approach for the lifted cuts (4.59)

input :

- Instance data $\Big(\mathcal{N}$, $\mathcal{M}$, $\mathcal{K}$, distances $D^0, ..., D^K$ and d_{ij} for each $i \in \{1...N\}$, $j \in \{1...M\}\Big)$
- Current (MP) solution $\bar{y}$ and a lower bound (LB, k^{LB})

output:

- $\tilde{k}$ index and the sub-problem objective value $\theta^{(SP)}$

1 $\theta^{(SP)} \leftarrow 0$
2 **for** $i \in \mathcal{N}$ **do**
3 $\quad \theta_i \leftarrow LB$
4 $\quad \beta_{ik} \leftarrow 1$
5 $\quad k = k^{LB}$
6 $\quad$ **while** $\beta_{ik} > 0$ **do**
7 $\quad\quad k \leftarrow k + 1$
8 $\quad\quad \beta_{ik} \leftarrow \beta_{ik} - \sum_{j:d_{ij}=D^{k-1}} \bar{y}_j$
9 $\quad\quad \theta_i \leftarrow \theta_i + (D^k - D^{k-1}) \max(0,\ \beta_{ik})$
10 $\quad\quad$ **if** $(\beta_{ik} > 0)$ **then**
11 $\quad\quad\quad \theta_i \leftarrow \theta_i + (D^k - D^{k-1})\beta_{ik}$
12 $\quad\quad\quad \tilde{k} \leftarrow k$
13 $\quad$ **if** $\theta_i > LB$ **then**
14 $\quad\quad$ Add $\theta \geq D^{\tilde{k}} - \sum_{j:d_{ij}<D^{\tilde{k}}} (D^{\tilde{k}} - \max\{d_{ij},\ LB\})y_j$ to (MP)
15 $\quad\quad$ **if** $\theta_i > \theta^{(SP)}$ **then**
16 $\quad\quad\quad \theta^{(SP)} \leftarrow \theta_i$
17 **return** $(\tilde{k}, \theta^{(SP)})$

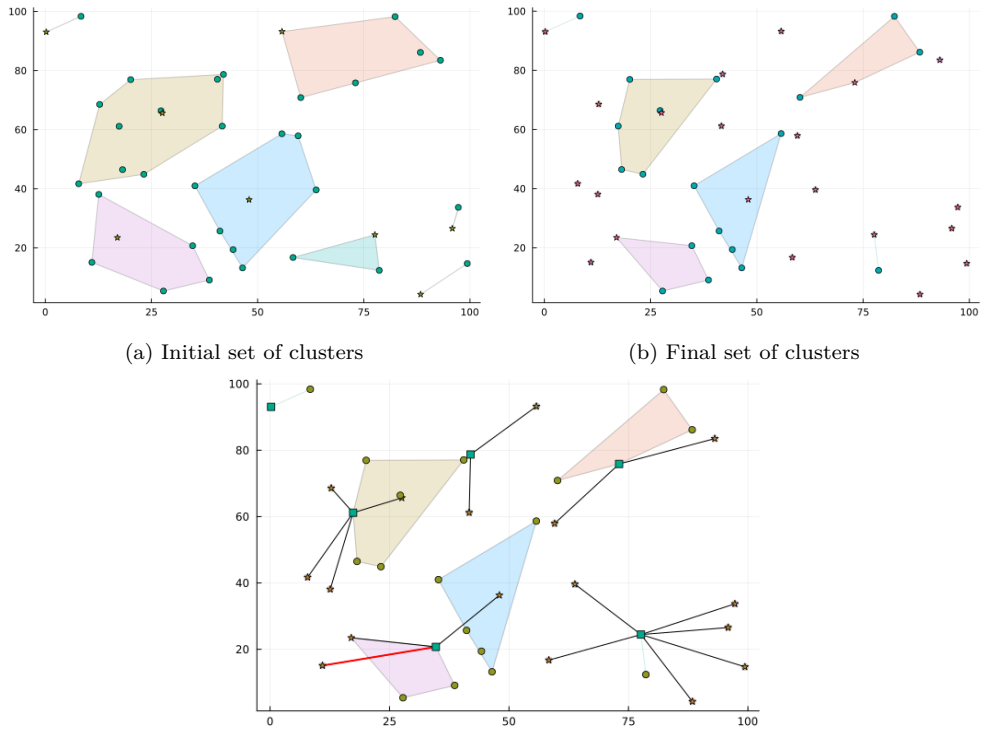
4.6 Clustering decomposition algorithm

We now propose a new exact solution method that takes advantage of the specific characteristics of the (PCP). As exploited by both Contardo et al. (2019) and Gaar and Sinnl (2022), the (PCP) can be solved by considering a subset of clients, and increasing this subset until the optimality is reached.

4.6.1 Clustering Procedure

The main idea of our clustering procedure is to efficiently solve the (PCP) by considering only the representatives of a set of the clusters of a partition of the clients. Once a solution of (PCP) is obtained, we check if it is optimal for the initial set of clients $i \in \mathcal{N}$. If not, the number of clusters considered is increased. To that end, we remove relevant clients from the clusters, each of them becoming a new cluster of size one.

Figure 4.2a shows the instance of 40 nodes considered in Chapter 1 with a partition of the clients into 8 clusters in which the representatives are marked as stars. Applying our algorithm, clients are sequentially removed from these clusters until the partition in Figure 4.2b is obtained. Consequently, only considering the associated representatives is sufficient to obtain the optimal solution presented in Figure 4.2c.



(a) Initial set of clusters

(b) Final set of clusters

(c) 6-center solution with clusters

Figure 4.2: Illustration of our clustering algorithm

More formally, let $\mathcal{C} = \{C_1, ..., C_T\}$ be the partition of the clients into T clusters and let r_t be the representative of cluster C_t. Let $y_{\mathcal{C}}$ be the set of sites opened in an optimal solution of the (PCP) on the set of representatives $I = \{r_t\}_{t=1}^{T}$ and let $v_{\mathcal{C}}$ be its objective value. Indeed, for each solution of a (PCP), a lower and an upper bound can be identified. The value $v_{\mathcal{C}}$ is a lower bound of the original problem since considering a subset of clients can only reduce the objective value. Moreover, the solution y_r can be evaluated on all the clients $i \in \mathcal{N}$, thus providing an upper bound.

To significantly reduce the computation time, we extend the notion of *dominated sites* introduced in Church (1984).

Definition 4.6.1. *A site $j_1 \in \mathcal{M}$ is dominated by another site $j_2 \in \mathcal{M}$ if $d_{ij_1} \geq d_{ij_2}$ for all clients $i \in I \subset \mathcal{N}$.*

Proposition 4.6.1. *There necessarily exists an optimal solution in which the opened sites are all non dominated.*

Proof. If a site j_1 dominated by j_2 is opened, then j_1 can be replaced by j_2 without deteriorating the value of the solution. □

Algorithm 9 outlines the general structure of our algorithm. To obtain an initial set of clusters (Step 1), we consider the well-known clustering heuristic *K-means*. It aims to minimize the sum of squared distances between data points and their designated cluster centers. The algorithm begins by randomly placing cluster centers, then assigning data points based on their proximity to these centers. After that, the centers are recalculated according to the assigned points, and the process is repeated until the stabilization of centers is achieved. The algorithm is run from different initializations and the best solution obtained is kept. K-means requires specifying the number of clusters beforehand. We consider $p + 2$ as the value of K.

For each cluster C_t, the client whose coordinates are the closest to the barycenter of all clients in the cluster is selected as the representative r_t. The advantage of this approach is that we do not need to compute the distances between all clients and all sites which would be prohibitive for large instances. Instead, we only compute the distances between the cluster representatives and the sites (Steps 2 and 3), and then identify the non-dominated sites (Step 4).

To improve the efficiency of the algorithm, we consider a two stage implementation. In the first stage, the linear relaxation of (PCP) is solved (Steps 8 to 12) while in the second stage we solve the integer (PCP). More precisely, in the first stage, a fractional solution y_c with an objective value of LB is obtained (Step 8). From y_c, we obtain an integer solution y_i by opening the sites associated with the p highest components of y_c (Step 9). To obtain more interesting

integer solutions, we apply the local search heuristic from Contardo et al. (2019) to solution y_i (Step 10 and 13). This enables to find a set of solutions S that cover within the same radius LB more non-representative clients and can potentially improve the upper bound UB. For each solution in S, we then try to identify non-representative clients whose allocation distance is greater than UB to create new clusters (Steps 11 and 14). This procedure for updating the clusters is summarized in Algorithm 10.

Algorithm 9: Clustering decomposition algorithm for (PCP)

input :

- Instance data $\left(\mathcal{N}, \mathcal{M}, p\right)$
- A lower bound LB and an upper bound UB

output:

- An optimal (PCP) solution y_{UB} and its value UB

1 Create the initial clusters $\mathcal{C} = \{C_1, ..., C_T\}$ and a set $I = \{r_t\}_{t=1}^{T}$ of their representatives
2 Compute the distances $d_{r_t j}$ between all representatives $r_t \in I$ and all the sites $j \in \mathcal{M}$
3 Compute the ordered distances $D^0, ..., D^{K_\mathcal{C}}$ between representatives I and all sites in M
4 Identify the set $\overline{J}$ of non-dominated sites for the representatives I
5 $(LB, UB) \leftarrow (-\infty, +\infty)$
6 **while** $LB < UB$ **do**
7 **while** *the size of* $|\mathcal{C}|$ *increases* **do**
8 $(y_c, LB) \leftarrow SolveContinousPCP(I, \overline{J})$ // First stage
9 $(y_i, \theta) \leftarrow getFeasibleSolution(y_c)$
10 $S \leftarrow findAlternativeSolutions(y_i, \theta)$
11 $(\mathcal{C}, I, \overline{J}, UB) \leftarrow updateClusters(\mathcal{C}, I, \overline{J}, \theta, S, UB)$
12 $(y_i, \theta, LB) \leftarrow SolveIntegerPCP(I, \overline{J})$ // Second stage
13 $S \leftarrow findAlternativeSolutions(y_i, \theta)$
14 $(\mathcal{C}, I, \overline{J}, UB) \leftarrow updateClusters(\mathcal{C}, I, \overline{J}, \theta, S, UB)$
15 **return** y_{UB}, UB

The clients in each cluster C_t are divided into four quadrants depending on whether they are located above, under, on the left or on the right of their representative r_t. For each solution $y_s \in S$ and each quadrant we remove from the cluster the client furthest from r_t that is not covered at distance UB by y_s. The use of quadrants enables to both limit the number of new clusters created and to add ones which positions are as diverse as possible. At this step, the optimal radius over all clients $i \in N$ is also computed and the distances and non-dominated sites are updated.

Algorithm 10: *updateClusters*$(C, I, \overline{J}, \theta, S, UB)$

```
1  for y_s ∈ S do
2      for C_t ∈ 𝒞 do
3          for quadrants of C_t do
4              for i ∈ C_t in decreasing order of distance to r_t do
5                  if ∃j ∈ y_s such that d_ij > θ then
6                      𝒞 ← 𝒞 ∪ {i},  I ← I ∪ {i}  // Update of clusters and representatives sets
7                      break        // Only 1 client removed form the cluster per quadrant
8      θ̄_s ← radius of solution y_s over all clients i ∈ 𝒩
9      if θ̄_s < UB then
10         UB ← θ̄_s
11 if UB has been updated then
12     Update the distances d_{r_t j}, D^0, ..., D^{K_C}, and the non-dominated sites J̄.
```

4.6.2 Solving the corresponding (PCP)

Algorithm 9 requires the solution of continuous and integer (PCP) (Steps 8 and 12) associated with the set of representatives I and non-dominated sites $\overline{J}$. In the computational experiments (Section 4.7), we have considered two approaches. The first one is based on our Benders decomposition presented in Section 4.5. The second one is based on a binary search of an associated set cover problem from Elloumi et al. (2004) and Contardo et al. (2019) previously described in Section 4.4.3. This latter approach also inspired the two stages of our Algorithm 9 and is also considered in Contardo et al. (2019).

4.6.3 Rounding the distances

The solution time of the (PCP) generally significantly depends on K, the number of distinct distances between the clients and the sites. To improve the performances, we propose the iterative Algorithm 11 in which the distances are rounded down less and less coarsely until an optimal solution is obtained.

Algorithm 11: solveByModuloClusters()

input :

- Instance data $\left(\mathcal{N}, \mathcal{M}, p\right)$
- A lower bound LB and an upper bound UB

output:

- An optimal (PCP) solution y and its value UB

1 $\mathcal{C} \leftarrow$ Partition of the clients

2 $\alpha \leftarrow$ Average number of digits in the distances between sites and representatives

3 **while** $\alpha \geq 0$ **do**

4 $\quad (LB, y, \mathcal{C}) \leftarrow$ Use Algorithm 9 to solve the (PCP) in which each distance d_{ij} is replaced by $10^{\alpha}\lfloor\frac{d_{ij}}{10^{\alpha}}\rfloor$ and $\mathcal{C}$ is used as the initial partition

5 $\quad \theta \leftarrow$ Radius of solution y without rounding the distances

6 $\quad UB \leftarrow \min(UB, \theta)$

7 $\quad \alpha \leftarrow \alpha - 1$

8 **return** (y, UB)

Note that since all the distances are rounded down at Step 4, the value LB obtained constitute a lower bound on the optimal solution. Moreover, the evaluation of the solution on all clients with the original distances provides an upper bound of at most $LB + 10^{\alpha}$ (Steps 5 and 6). This enables to further reduce the number of distinct distances considered at the next iteration of Step 4.

Finally, since the clients removed from their cluster at a given iteration are likely to be relevant for the subsequent iterations we only cluster the clients once at the beginning of Algorithm 11. The same partition $\mathcal{C}$ is then both used as the initial partition and updated within Algorithm 9 (Step 4).

4.7 Computational experiments

In this section we present the experimental results of the Benders decomposition and of the clustering algorithm. Our study was carried out on an Intel XEON W-2145 processor 3,7 GHz, with 16 threads, but only 1 was used, and 256 GB of RAM. IBM ILOG CPLEX 20.1 was used as MILP solver for all the methods. Particularly, the Benders decomposition method presented in Section 4.5 was implemented in C++ while the clustering and rounding methods presented in Section 4.6 were implemented in Julia.

4.7.1 Second Benders decomposition performance

As the p-median problem in Section 3.3.5, we consider a two-phase Benders decomposition implementation. In the first phase, the master problem is relaxed, and in the second one, the integrality of variables of the master problem are restored and it is solved in a *branch-and-cut* framework. To illustrate the impact of the different Benders cuts, we have noted each of the separation approaches as follows:

- *sck*: separation of a single cut involving different $i^*_{(k)}$ indexes depending on k (i.e., use of Algorithm 6 with the non-lifted inequalities (4.52)).
- *sckL*: separation of a lifted single cut involving different $i^*_{(k)}$ indexes depending on k (i.e., use of Algorithm 6 with the lifted inequalities (4.58)).
- $sc\tilde{k}$: separation of a single cut involving only the index $i^*_{(\tilde{k})}$ (i.e., use of Algorithm 7 with the non-lifted inequalities (4.53)).
- $sc\tilde{k}L$: separation of a lifted single cut involving only the index $i^*_{(\tilde{k})}$ (i.e., use of Algorithm 7 with the lifted inequalities (4.59)).
- $mc\tilde{k}$: separation of a cut involving an index $i^*_{(\tilde{k})}$ for each client $i \in \mathcal{N}$ (i.e., use of Algorithm 8 with the non-lifted inequalities (4.53).
- $mc\tilde{k}L$: separation of a lifted cut involving an index $i^*_{(\tilde{k})}$ for each client $i \in \mathcal{N}$ (i.e., use of Algorithm 8 with the lifted inequalities (4.59)).

Table 4.3 presents the results obtained in five ORLIB instances. For both phases, we present the optimality gap at the end of the phase to the best feasible solution (*gap*), the number of iterations (*iters*), the number of cuts added (*cuts*) and the cpu time ($t[s]$). For the first phase, we present the corresponding lower bound (LB), while for the second phase we present the upper bound (UB). A time limit ($TL3$) of 500 seconds has been considered. The last column shows the total cpu time of both phases.

Firstly, we can note the great effect of the lifting procedure of Gaar and Sinnl (2022) adapted to our Benders decomposition, which allows to increase the lower bounds in the first phase, thus reducing the iterations and cuts on both phases in most instances for the three approaches. Although Algorithm 6 leads to higher LB, it is less efficient in terms of computational time. This approach without the lifting cannot solve to optimality the instances within the time limit. The same applied to Algorithm 7 even though it is better in most instances. In contrast, the multi-cut approach of Algorithm 8 is much faster than the others without the lifting. Although this multi-cut implementation does not present the best total time, this approach in Gaar and Sinnl (2022) can be much faster for the same instances. Consequently, implementation improvements are required in Algorithm 6 in order to take better advantage of the strength of its relaxation equal to that of $(F2)$ as opposed to the other which relaxation is equal to that of $(F1)$, as discussed in the Section 4.5.2.

Instance			Algorithm	Phase 1					Phase 2					Total
name	$N = M$	p		LB	*gap*	*iters*	*cuts*	$t[s]$	UB	*gap*	*iters*	*cuts*	$t[s]$	time[s]
pmed03	100	10	sck	**81**	34%	1260	1260	19.96	96	8%	73	73	TL3	TL3
			$sc\tilde{k}$	62	63%	99	99	0.26	93	0%	6973	6973	5.70	5.96
			$mc\tilde{k}$	62	95%	7	247	0.02	93	0%	40	1065	2.41	**2.43**
			$sckL$	**93**	2%	651	651	5.07	93	0%	1	1	0.12	5.19
			$sc\tilde{k}L$	86	20%	109	109	0.12	93	0%	62	62	0.43	**0.54**
			$mc\tilde{k}L$	46	168%	2	90	0.01	93	0%	32	588	0.60	0.61
pmed06	200	5	sck	**75**	22%	669	669	11.20	84	2%	90	90	TL3	TL3
			$sc\tilde{k}$	63	45%	71	71	0.21	84	0%	5670	5670	16.14	16.36
			$mc\tilde{k}$	63	53%	5	324	0.03	84	0%	5	285	1.16	**1.19**
			$sckL$	**82**	7%	161	161	0.40	84	0%	2	2	7.07	7.47
			$sc\tilde{k}L$	79	15%	72	72	0.07	84	0%	11	11	0.34	**0.41**
			$mc\tilde{k}L$	56	72%	2	195	0.02	84	0%	30	529	1.13	1.15
pmed11	300	5	sck	**53**	15%	379	379	4.75	59	2%	105	105	TL3	TL3
			$sc\tilde{k}$	46	33%	47	47	0.12	59	0%	5074	5074	6.45	6.56
			$mc\tilde{k}$	46	44%	5	412	0.04	59	0%	8	690	1.68	**1.72**
			$sckL$	**58**	3%	101	101	0.20	59	0%	3	3	0.60	0.80
			$sc\tilde{k}L$	56	7%	60	60	0.05	59	0%	121	121	0.23	**0.28**
			$mc\tilde{k}L$	41	62%	2	295	0.02	59	0%	16	246	1.76	1.79
pmed22	500	10	sck	**34**	23%	1004	1004	39.93	39	10%	89	89	TL3	TL3
			$sc\tilde{k}$	29	44%	111	111	0.56	39	12%	3530	3530	TL3	TL3
			$mc\tilde{k}$	29	53%	4	675	0.06	39	7%	44	2513	TL3	TL3
			$sckL$	**37**	5%	399	399	10.15	38	0%	16	16	128.09	138.24
			$sc\tilde{k}L$	35	17%	102	102	0.18	38	0%	89	89	27.55	**27.73**
			$mc\tilde{k}L$	26	66%	2	490	0.03	38	0%	29	664	31.81	31.84
pmed26	600	5	sck	**34**	14%	334	334	5.83	38	4%	15	15	TL3	TL3
			$sc\tilde{k}$	30	36%	60	60	0.23	38	0%	3296	3296	113.42	**113.65**
			$mc\tilde{k}$	30	42%	4	704	0.09	38	0%	15	835	246.68	246.77
			$sckL$	**37**	3%	112	112	0.32	38	0%	6	6	9.49	9.81
			$sc\tilde{k}L$	36	6%	54	54	0.08	38	0%	168	168	0.52	**0.60**
			$mc\tilde{k}L$	28	56%	2	595	0.07	38	0%	16	179	7.71	7.78

Table 4.3: Results on ORLIB instance of the performance between Benders decompositions according to the cut separation approach. TL3=500s.

4.7.2 Clustering performance

Different methods can be considered to solve the (PCP) encountered in our clustering method presented in Algorithm 10. The binary search algorithm from Elloumi et al. (2004), while not being the most efficient to solve large (PCP) problems by itself, appeared to be the most efficient within our clustering approach. Indeed, this approach provided better results than when our clustering algorithms was combined with the branch-and-cut algorithm from Gaar and Sinnl (2022). Consequently, in the following, all the (PCP) problems of our clustering approach are solved using the binary search algorithm.

To enable a fair comparison, we were able to run the methods from Gaar and Sinnl (2022) and Contardo et al. (2019) with CPLEX on the same computer as for our clustering algorithm. In particular, the code of Gaar and Sinnl (2022) was provided to us as an executable program with a defined time limit of 1800 seconds $(TL4)$. Therefore, we have set the same time limit for the two other methods. The results are summarized in the Tables 4.4 and 4.5. We present the lower bound (LB), the value (UB) of the best solution found (in **bold** if it is optimal), the relative gap between these bound (gap), and the cpu-time in seconds $(t[s])$.

Our method outperforms Gaar and Sinnl (2022) and Contardo et al. (2019) on most TSP instances in Table 4.4, only Gaar and Sinnl (2022) have better solution times for small values of p. On the other hand, Contardo et al. (2019) outperform Gaar and Sinnl (2022) in certain instances, as already presented Gaar and Sinnl (2022). For large TSP instances in Table 4.5, both methods can be faster than us for small values of p. However, when the time limit is reached, better optimality gaps are obtained in most instances. The scalability of our method is based on the iterative rounding of the distances which could generate larger optimality gap in the first iterations. Consequently, our method could obtain better comparison results with a longer time limit.

Instance		*UB*			*LB*			*gap*			*t*[*s*]		
name	*p*	Clus.	Gaar	Cont.	Clus.	Gaar	Cont.	Clus.	Gaar	Cont.	Clus.	Gaar	Cont.
fl1400	2	**1193**	**1193**	**1193**	1193	1193	1193	0%	0%	0%	0.1	**0.1**	5.7
	3	**909**	**909**	**909**	909	909	909	0%	0%	0%	**0.1**	0.1	0.5
	5	**662**	**662**	**662**	662	662	662	0%	0%	0%	0.1	**0.1**	0.6
	10	**389**	**389**	**389**	389	389	389	0%	0%	0%	**0.1**	0.1	0.8
	15	**312**	**312**	**312**	312	312	312	0%	0%	0%	0.2	**0.1**	0.9
	20	**246**	**246**	**246**	246	246	246	0%	0%	0%	**0.2**	0.2	1.5
	50	**87**	**87**	**87**	87	87	87	0%	0%	0%	0.4	**0.3**	3.7
u1817	2	**1061**	**1061**	**1061**	1061	1061	1061	0%	0%	0%	0.2	**0.1**	1.2
	3	**895**	**895**	**895**	895	895	895	0%	0%	0%	0.2	**0.1**	0.6
	5	**715**	**715**	**715**	715	715	715	0%	0%	0%	**0.2**	0.7	2.8
	10	**458**	**458**	**458**	458	458	458	0%	0%	0%	**4**	20	19
	15	**359**	**359**	**359**	359	359	359	0%	0%	0%	**13**	837	110
	20	**309**	310	**309**	309	306	309	0%	1%	0%	**92**	TL4	916
	50	**185**	198	203	185	180	180	0%	10%	13%	**1001**	TL4	TL4
d2103	2	**1787**	**1787**	**1787**	1787	1787	1787	0%	0%	0%	0.4	**0.1**	0.9
	3	**1420**	**1420**	**1420**	1420	1420	1420	0%	0%	0%	0.2	**0.1**	1.3
	5	**1032**	**1032**	**1032**	1032	1032	1032	0%	0%	0%	**0.2**	0.3	1.6
	10	**668**	**668**	**668**	668	668	668	0%	0%	0%	**1**	4	8
	15	**532**	532	**532**	532	528	532	0%	1%	0%	**12**	TL4	54
	20	**434**	**434**	**434**	434	434	434	0%	0%	0%	**15**	20	50
	50	**254**	257	277	254	243	243	0%	6%	14%	**556**	TL4	TL4
fl3795	2	**1147**	**1147**	**1147**	1147	1147	1147	0%	0%	0%	0.1	**0.1**	0.5
	3	**938**	**938**	**938**	938	938	938	0%	0%	0%	0.3	**0.2**	0.7
	5	**587**	**587**	**587**	587	587	587	0%	0%	0%	**0.3**	0.4	1.0
	10	**413**	**413**	**413**	413	413	413	0%	0%	0%	**0.5**	0.8	2.5
	15	**312**	**312**	**312**	312	312	312	0%	0%	0%	0.6	**0.5**	3.6
	20	**269**	**269**	**269**	269	269	269	0%	0%	0%	**0.9**	1.5	7.1
	50	**101**	**101**	**101**	101	101	101	0%	0%	0%	**3**	4	37
usa13509	2	**175,750**	**175,750**	**175,750**	175,750	175,750	175,750	0%	0%	0%	1	**1**	8
	3	**134,489**	**134,489**	**134,489**	134,489	134,489	134,489	0%	0%	0%	2	**1**	3
	5	**103,671**	**103,671**	**103,671**	103,671	103,671	103,671	0%	0%	0%	**3**	4	14
	10	**67,075**	**67,075**	**67,075**	67,075	67,075	67,075	0%	0%	0%	**26**	185	133
	15	**52,178**	58,306	56,372	52,178	51,484	52,095	0%	13%	8%	**165**	TL4	TL4
	20	44,994	55,373	55,375	44,500	43,724	43,506	1%	27%	27%	TL4	TL4	TL4
	50	30,000	35,363	29,770	20,000	25,424	24,282	50%	39%	23%	TL4	TL4	TL4
sw24978	2	**4960**	**4960**	**4960**	4960	4960	4960	0%	0%	0%	4	**1**	4
	3	**4120**	**4120**	**4120**	4120	4120	4120	0%	0%	0%	5	**2**	4
	5	**3022**	**3022**	**3022**	3022	3022	3022	0%	0%	0%	**6**	6	13
	10	**2061**	**2061**	**2061**	2061	2061	2061	0%	0%	0%	**19**	50	96
	15	**1637**	**1747**	**1637**	1637	1629	1637	0%	7%	0%	**250**	TL4	655
	20	**1421**	1683	1471	1421	1406	1409	0%	20%	4%	**1135**	TL4	TL4
	50	998	1114	1026	800	812	767	25%	37%	34%	TL4	TL4	TL4
ch71009	2	**19,988**	**19,988**	**19,988**	19,988	19,988	19,988	0%	0%	0%	29	**3**	6
	3	**13,292**	**13,292**	**13,292**	13,292	13,292	13,292	0%	0%	0%	27	**8**	17
	5	**10,944**	**10,944**	**10,944**	10,944	10,944	10,944	0%	0%	0%	**42**	110	100
	10	**7183**	**7183**	**7183**	7183	7183	7183	0%	0%	0%	**176**	1180	744
	15	5699	6099	6091	5650	5592	5588	1%	9%	9%	TL4	TL4	TL4
	20	5000	6016	5223	4000	4678	4613	25%	29%	13%	TL4	TL4	TL4
	50	3000	3740	3936	2000	2653	2441	50%	41%	61%	TL4	TL4	TL4

Table 4.4: Results in TSP instances of the performance between the algorithms based on the clustering approach. TL4=1800s.

Instance		UB			LB			gap			$t[s]$		
name	p	Clus.	Gaar	Cont.	Clus.	Gaar	Cont.	Clus.	Gaar	Cont.	Clus.	Gaar	Cont.
pla85900	2	**436,008**	**436,008**	**436,008**	436,008	436,008	436,008	0%	0%	0%	41.3	**11.7**	40.9
	3	**399,677**	**399,677**	**399,677**	399,677	399,677	399,677	0%	0%	0%	**56.8**	67.0	267.7
	5	**269,544**	**269,544**	287,961	269,544	269,544	268308	0%	0%	7%	**123**	631	TL4
	10	**180,496**	201,814	197,009	180,496	177,841	173,506	0%	13%	14%	**622**	TL4	TL4
	15	160,000	178,702	179,921	140,000	138,323	132,202	14%	29%	36%	TL4	TL4	TL4
	20	130,000	163,483	150,621	120,000	116,355	109,026	8%	41%	38%	TL4	TL4	TL4
sra104815	2	**908**	**908**	**908**	908	908	908	0%	0%	0%	37	**8**	24
	3	**688**	**688**	**688**	688	688	688	0%	0%	0%	48	**15**	46
	5	**508**	**508**	**508**	508	508	508	0%	0%	0%	64	**61**	143
	10	**354**	410	384	354	349	352	0%	17%	9%	**575**	TL4	TL4
	15	**277**	351	312	277	274	274	0%	29%	14%	**1645**	TL4	TL4
	20	240	293	272	230	229	225	4%	28%	21%	TL4	TL4	TL4
usa115475	2	**17,745**	**17,745**	**17,745**	17,745	17,745	17,745	0%	0%	0%	74	**6**	16
	3	**13,530**	**13,530**	**13,530**	13,530	13,530	13,530	0%	0%	0%	82	**10**	29
	5	**10,414**	**10,414**	**10,414**	10,414	10,414	10,414	0%	0%	0%	172	**125**	301
	10	**6736**	6811	6824	6736	6726	6735	0%	1%	1%	**459**	TL4	TL4
	15	5600	6363	5681	5200	5140	5044	8%	24%	13%	TL4	TL4	TL4
	20	4998	5639	5223	4000	4275	4058	25%	32%	29%	TL4	TL4	TL4
ara238025	2	**1484**	**1484**	**1484**	1484	1484	1484	0%	0%	0%	187	**23**	52
	3	**1166**	**1166**	**1166**	1166	1166	1166	0%	0%	0%	193	**31**	72
	5	**855**	**855**	**855**	855	855	855	0%	0%	0%	443	**178**	378
	10	**552**	561	569	552	550	549	0%	2%	4%	**1491**	1843	1835
	15	499	508	474	410	435	423	22%	17%	12%	TL4	TL4	TL4
	20	400	479	450	300	360	348	33%	33%	29%	TL4	TL4	TL4
lra498378	2	**5888**	**5888**	**5888**	5888	5888	5888	0%	0%	0%	1099	**104**	195
	3	**4995**	**4995**	**4995**	4995	4995	4995	0%	0%	0%	847	**171**	234
	5	**3259**	**3259**	**3259**	3259	3259	3259	0%	0%	0%	1439	**317**	785
	10	2299	2315	2362	2200	2199	2119	5%	5%	11%	TL4	TL4	TL4
	15	1998	2078	2076	1000	1682	1592	100%	24%	30%	TL4	TL4	TL4
	20	1500	1714	1761	1400	1353	1216	7%	27%	45%	TL4	TL4	TL4
lrb744710	2	1950	**1930**	**1930**	1900	1930	1930	3%	0%	0%	TL4	**78**	80
	3	**1702**	**1702**	**1702**	1702	1702	1702	0%	0%	0%	1780	**189**	298
	5	1199	**1170**	1175	1100	1170	1168	9%	0%	1%	TL4	**1790**	TL4
	10	800	916	980	700	760	741	14%	21%	32%	TL4	TL4	TL4
	15	700	697	762	600	602	558	17%	14%	37%	TL4	TL4	TL4
	20	600	638	647	500	493	451	20%	29%	43%	TL4	TL4	TL4

Table 4.5: Results in large TSP instances of the performance between the algorithms based on the clustering approach. TL4=1800s.

4.8 Conclusions

The five main (PCP) formulations lead to performances that can vary significantly depending on the solver parameters, such as the presolve. Among them, formulations $(F4)$ and $(F2)$ appear to be the most efficient. However, $(F2)$ performs better than $(F4)$ when the presolve is deactivated, as it has fewer redundant constraints.

Two Benders decompositions have been studied for formulation $(F2)$ depending on the set of variables to be relaxed. When the decision variables associated with the location of the facilities are relaxed, we obtain the set cover problem as primal sub-problems and the maximal stable set problem as dual sub-problems. This highlights a close relationship with one of the classical exact solution methods of (PCP), in which set cover sub-problems are also considered within in a binary search. Since the master problem is easy to solve, the Benders decomposition can be seen as a linear search on the distances. Indeed, for each evaluated distance, we check if there is a feasible solution of p sites and we go to the next distance at the next iteration. However, this implementation has no better results than this classical binary search. On the other hand, when the variables associated with the allocation of the clients to the open facilities are relaxed, the sub-problem is easy to solve. We show that the decomposition on $(F2)$ allows to have new Benders cuts that are stronger than the previously known ones. We also highlight the relation with the Benders decomposition from formulation $(F1)$ in Gaar and Sinnl (2022). All approaches have been tested, and although our Benders decomposition from $(F2)$ allows us to obtain better linear relaxation bounds, with our current implementation, it takes more computation time than the multi-cut approach from $(F1)$ implemented in Gaar and Sinnl (2022).

Finally, an exact method based on a clustering of the clients has been proposed. It solves the (PCP) iteratively for a set of representative clients of the clusters that is updated when necessary. A first stage of the relaxed (PCP) problem is considered followed by a second stage to solve the integer (PCP). Moreover, to reduce the number of distinct distances values, we also consider an exact iterative distance rounding procedure. All these implementation improvements lead to competitive results with state-of-the-art methods of Gaar and Sinnl (2022) and Contardo et al. (2019) on TSP instances.

The main perspective of the Benders Decomposition of (PCP) on $(F2)$ is to improve our implementation to take advantage of the strength of the new cuts in terms of lower bound in contrast to multi-cut approach. Meanwhile, the results of our clustering algorithm suggest that it can be applied to other discrete location problems or other types of instances. Likewise, to achieve a good implementation that combines both approaches.

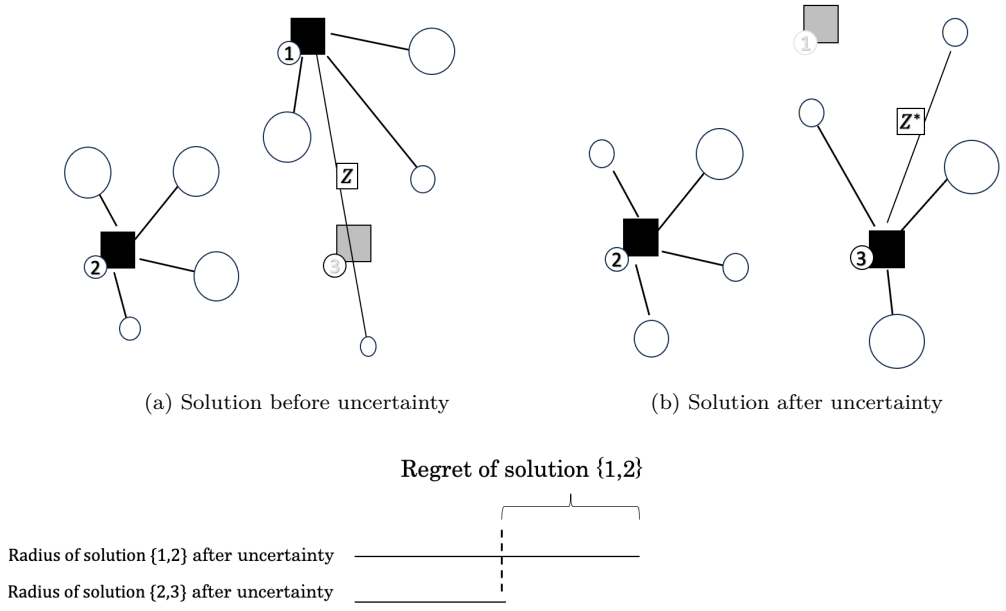
Chapter 5

MILP formulations and exact algorithms for the robust two-stage p-center problem

5.1 Introduction

The p-center problem (PCP) under uncertainty is well studied in the literature (Çalık et al. (2019b)). This problem arises when parameters, such as demands or distances between the demand nodes and the available sites, vary across time or when their exact value is uncertain. In robust optimization uncertainty is generally represented by parameters which can take any value in an uncertainty set. Each realization of the uncertainty set is called a *scenario*. The most classical sets are the box, the ellipsoidal and the budgeted uncertainty sets (see e.g., Ben-Tal et al. (2009); Bertsimas and Sim (2004); Du and Zhou (2018); Paul and Wang (2019)).

A classical objective function in robust optimization is the regret one have for initial decision once the uncertainty is revealed. Figure 5.1 illustrates how the optimal solution of a problem can change with the values of its parameters. In this example, we consider a p-center problem in which each client has an uncertain demand. The demand value is represented by the size of the circles. Figure 5.1a represents the value of the initial parameters and the corresponding optimal solution which consists in opening sites 1 and 2. Once the uncertainty is revealed, the parameters take the value represented in Figure 5.1b. We can see that this modifies the optimal solution which is now to open sites 2 and 3. In this context, the regret corresponds to the difference between the objective values of the two solutions when the parameters take their real values represented in Figure 5.1b.



(a) Solution before uncertainty (b) Solution after uncertainty

Figure 5.1: Example of the regret of optimal (RPCP) solutions before and after uncertainty

The incorporation of uncertainty in (PCP) has important applications in emergency logistics problems, where a prompt response to the urgent need for relief is required in affected areas immediately following a disaster (such as earthquakes, tsunamis, landslides, among others). The consequences of these disasters make it challenging to precisely estimate the demand for relief materials or the travel times between relief centers and affected locations (Sheu (2007)).

Two approaches can be considered depending on whether the demand node allocations to the centers are made before (see e.g, Averbakh and Berman (1997), Lu (2013)) or after (see e.g, Du et al. (2020); Demange et al. (2020)) the uncertainty is revealed. The first case corresponds to single-stage problems while the second case leads to two-stage problems in which the demand node allocations are recourse variables. In this context, most of the works have been focused on the investigation of integer programming modeling and heuristic resolution approaches (see e.g., Baron et al. (2011); Hasani and Mokhtari (2018); Paul and Wang (2015); Trivedi and Singh (2017, 2019)).

Contribution

To the best of our knowledge this work is the first one to study the exact solution of the two-stage robust weighted vertex p-center problem $(RPCP_2)$, with $p > 1$, in which the uncertainty on the node demands and distances are modeled by box uncertainty sets. We present the robust reformulation of this problem based on five MILP formulations of the (PCP) from the literature.

We prove that a finite subset of scenarios from the infinite box uncertainty set can be considered without losing optimality. We use this result to propose a *column-and-constraint generation* algorithm ($C\&CG$) and a *branch-and-cut* algorithm ($B\&C$) for the exact solution of ($RPCP_2$). We highlight how these algorithms can be adapted to the single-stage problem ($RPCP_1$). Finally, we show their efficiency on randomly generated instances. To illustrate the usefulness of the proposal, we consider the case study presented in Lu (2013) which is based on an earthquake that hit central Taiwan in 1999.

The rest of the chapter is organized as follows: Section 5.2 presents the literature review of the deterministic and robust versions of the (PCP). Section 5.3 describes the robust two-stage problem ($RPCP_2$), proves how to reduce the number of considered scenarios, and introduces the five MILP formulations as well as the ($C\&CG$) and ($B\&C$) algorithms. Section 5.4 presents the computational results. Finally, conclusions and research perspectives are drawn in Section 5.6.

5.2 Literature review

The p-center problem was introduced by Hakimi (1965), who presented and solved the absolute 1-center problem on a graph. In the *absolute* p-center problem, the centers can be located either on the edges or the vertices of the graph. Later, Minieka (1970) extended the problem to the case $p > 1$ and proposed a method to restrict the continuous set of candidate centers to a discrete set of points, without losing optimality. Since then, the problem was commonly referred to as the *vertex* p-center problem or directly as the p-center problem. Several formulations, solution methods, and variants of this problem have been presented. A comprehensive review of the applications and solution methods of the p-center problem can be found in Çalık et al. (2019b). This section focuses on the deterministic p-center problem formulations and then on its robust counterparts.

5.2.1 MILP formulations of the deterministic weighted p-center

Let $\mathcal{M}$ be the set of available sites, and $\mathcal{N}$ be the set of demand nodes. We denote here as t_{ij} the distance (or travel time) between any possible pair of demand node $i \in \mathcal{N}$ and site $j \in \mathcal{M}$. Each demand node $i \in \mathcal{N}$ faces a demand d_i and must be allocated to a single center j.

All the deterministic formulations of the p-center problem have already been presented in Section 4.2.1. These are the classic formulation ($F1$) of Daskin (1996) with the (x, y) variables, the alternative one ($F2_{init}$) of Elloumi et al. (2004) with (y, z) variables, the improvement of this last one ($F2$) and the compact one ($F3$) with (y, r) variables both introduced in Ales and Elloumi (2018), formulation ($F4$) presented Calik and Tansel (2013) with the (y, u) variables,

and formulation $(F5)$ of Gaar and Sinnl (2022) with (y, θ) variables. To model the weighted vertex p-center, the original parameter of distances d_{ij} in these formulations can be replaced here by the product of demand d_i and distance t_{ij}.

As also mentioned in Section 4.2.1, $(F4)$ is the fastest formulation on average closely followed by $(F2)$. Nevertheless, as will be seen in Section 5.4 that the best formulations for the deterministic problem are not necessarily the best for the robust problem.

5.2.2 Uncertainty representation and solution methods

The location of facilities is a long-term decision which takes into account parameters such as demands or distances between demand nodes and facilities. Since these parameters are likely to vary, several models have been developed to study facility location problems under uncertainty. Stochastic optimization and robust optimization are the two main approaches to address uncertainty. The Section 2.3 presents a general introduction and references to optimization under uncertainty. Snyder (2006) and Laporte et al. (2019a) present a review of the literature on stochastic and robust facility location problems.

In robust optimization, box, budgeted, ellipsoidal and discrete uncertainty sets are commonly considered (see e.g. Ben-Tal et al. (2009); Baron et al. (2011); Du and Zhou (2018); Paul and Wang (2019, 2015); Snyder (2006)). Since most robust facility location problems are generally harder to solve than their deterministic counterparts, heuristic approaches have taken precedence over exact solution methods (Laporte et al. (2019a)). Most robust facility location problems based on discrete uncertainty sets deal with generalizations of the p-median problem, focusing exclusively on analytical results or approximated polynomial-time algorithms (see e.g. Serra and Marianov (1998); Hasani and Mokhtari (2018)).

The presence or absence of recourse variables, the variables which are fixed once the uncertainty is revealed, has a great influence on the mathematical formulation of the problem. A single-stage problem can be considered when there is no recourse variables while a two-stage is required otherwise. Two-stage models are usually very difficult to solve (Ben-Tal et al. (2009)). When the second stage problem is a linear program, Benders decomposition method can be used to seek optimal solutions (Bertsimas et al. (2013); Rahmaniani et al. (2017)).

Zeng and Zhao (2013) develop another exact solution method, the column-and-constraint generation ($C\&CG$) algorithm (also called row-and-column or scenario generation), which has performed better on different problems including facility location problems (see e.g. An et al. (2014); Chan et al. (2018)).

Several robust variants of (PCP) with either a single stage or two stages have been considered. For example, Averbakh and Berman (1997) consider the weighted p-center problem on a transportation network with uncertain node weights. They minimize the regret of the worst-case scenario and show that the problem can be solved through a number of particular weighted p-center problems. Averbakh and Berman (2000) consider a box uncertainty set for the weighted 1-center problem on a network with uncertainty node weights and edge lengths. Each uncertain parameter is assumed to be random with an unknown distribution. They present a polynomial algorithm to find the robust solution for the problem on a tree. Lu and Sheu (2013) consider the single-stage weighted vertex p-center with uncertain edge lengths using box uncertainty sets. They consider the single-stage robust problem ($RPCP_1$), and prove that it is sufficient to consider a discrete subset of scenarios, and propose a simulated annealing heuristic to solve the problem. In Lu (2013), they extend this research to the weighted vertex p-center with uncertain nodal weights and edge lengths using also box uncertainty sets. Du and Zhou (2018) apply a single-stage approach to a p-center problem based on a multiple allocation strategy (i.e., it is allowed the allocation of a client to several sites), and three types of symmetric uncertainty sets over units costs: box uncertainty, ellipsoidal uncertainty, and cardinality-constrained uncertainty, where a symmetric interval is defined as an interval where the lower and upper bounds are equidistant from the center. Du et al. (2020) propose a two-stage robust model for a reliable facility location problem, i.e., when some facilities can be disrupted and the demand nodes can be reallocated to another available facility. They consider uncertain demand and cost, and propose three solution methods: a linear reformulation, a Benders dual cutting planes method, and a *column-and-constraint generation* method. Demange et al. (2020) introduce the robust p-center problem under pressure motivated by the context of locating shelters for evacuation in case of wildfires, where the uncertainty is in the available network connections. They present a MILP formulation and a decomposition scheme to solve it.

Other robust facility location problems have been studied in the literature, mostly considering uncertainty in customer demand and/or facility disruption. Their solution methods are carried out by dualization techniques or column-and-constraint generation algorithms (Nikoofal and Sadjadi (2010); An et al. (2014); Cheng et al. (2021)). A detailed literature review is presented in Snyder (2006).

This research focuses on the exact solution of the two-stage weighted vertex p-center problem ($RPCP_2$), with $p > 1$, with uncertainty in both nodal weights and edge lengths, as shown in the last row of Table 5.1. We also show how these algorithms can be adapted to solve the single-stage problem ($RPCP_1$).

	Allocation level		Decision level		Number of centers		Uncertainty			Messure of robustness		Solution approach	
Article	Single	Multiple	Single-stage	Two-stage	$p=1$	$p>1$	Distance	Demand	Centers	Regret	Worst Case Value	Heuristic	Exact
Averbakh and Berman (1997)	x		x			x		x		x			x
Averbakh and Berman (2000)	x		x		x		x	x		x			x
Lu and Sheu (2013)	x		x			x	x			x		x	
Lu (2013)	x		x			x	x	x		x		x	
Du and Zhou (2018)		x	x			x	x			x			x
Du et al. (2020)	x			x		x			x	x			x
Demange et al. (2020)	x		x			x			x		x		x
This proposal	**x**			**x**		**x**	**x**	**x**		**x**			**x**

Table 5.1: A summary of related work.

5.3 Robust weighted vertex p-center problem

We first define the $(RPCP_2)$. We then prove that it is sufficient to consider a subset of the infinite scenarios in the box uncertainty set to obtain an optimal solution of the problem. Finally, the $(C\&CG)$ and $(B\&C)$ algorithms for the exact solution of both $(RPCP_2)$ and $(RPCP_1)$ are presented.

5.3.1 Problem definition

Following Lu (2013), we consider that the node demands and distances can take any value in a box uncertainty set. More precisely, the demand d_i of demand node $i \in \mathcal{N}$ is assumed to be in $[d_i^-, d_i^+]$ where $0 \leq d_i^- \leq d_i^+$, while the distance t_{ij} between station $i \in \mathcal{N}$ and site $j \in \mathcal{M}$ takes its value in $[t_{ij}^-, t_{ij}^+]$ where $0 \leq t_{ij}^- \leq t_{ij}^+$.

Let $W \subset \mathbb{R}^{N+M\times N}$ be the Cartesian product of intervals $[d_i^-, d_i^+]$ and $[t_{ij}^-, t_{ij}^+]$ for each $i \in \mathcal{N}$ and $j \in \mathcal{M}$. Let $\Omega = \{y \in \{0,1\}^M \mid \sum_{j\in\mathcal{M}} y_j = p\}$ be the set of vectors representing p located facilities and let $J_y = \{j \in \mathcal{M} \mid y_j = 1\}$ be the set of located facilities for vector $y \in \Omega$. For a given scenario $w \in \mathcal{W}$ let d_i^w and t_{ij}^w respectively be the demand of node $i \in \mathcal{N}$ and the distance between demand node i and site $j \in \mathcal{M}$ in scenario w.

In $(RPCP_2)$, the demand nodes are allocated after the uncertainty is revealed. This corresponds to a two-stage approach in which the location of centers is fixed at the first stage and the demand node allocations are the recourse decisions of the second stage. Consequently, the radius associated with $y \in \Omega$ when scenario $w \in \mathcal{W}$ occurs is:

$$Z(w,y) = \max_{i\in\mathcal{N}} \left\{ \min_{j\in J_y} d_i^w t_{ij}^w \right\} \tag{5.1}$$

which represents an optimal allocation of demand nodes in scenario w when sites J_y are located.

Let $y^*(w) \in \Omega$ be a vector such that the location of $J_{y^*(w)}$ leads to an optimal radius for the deterministic p-center problem in which the uncertain data takes value $w \in \mathcal{W}$. We define the *robust deviation* of $y \in \Omega$ for scenario w as:

$$DEV(w, y) = Z(w, y) - Z(w, y^*(w)) \tag{5.2}$$

It is a non-negative value corresponding to the increase in radius incurred when locating centers J_y rather than $J_{y^*(w)}$ in scenario w. The following lemma allows us to characterize situations where the deviation is zero.

Lemma 5.3.1. *Let $y \in \Omega$ and let w be a scenario. Let $(i, j) \in \mathcal{N} \times J_y$ be such that $Z(w, y) = d_i^w t_{ij}^w$, i.e., (i, j) allow to reach the optimal radius $Z(w, y)$. Let $\overline{j} = \underset{j \in J_{y^*(w)}}{\arg\min}\, t_{ij}^w$, i.e., node i is optimally assigned to site $\overline{j}$ when sites set $J_{y^*(w)}$ of solution $y^*(w)$ are opened. It holds that if $\overline{j} \in J_y$ then $DEV(w, y) = 0$.*

Proof $DEV(w, y) \geq 0$ by definition. Suppose $DEV(w, y) > 0$, i.e., $Z(w, y) > Z(w, y^*(w))$. As i is assigned to $\overline{j}$ for solution $y^*(w)$, $Z(w, y^*(w)) \geq d_i^w t_{i\overline{j}}^w$. Therefore, $Z(w, y) = d_i^w t_{ij}^w > d_i^w t_{i\overline{j}}^w$ and then $t_{ij}^w > t_{i\overline{j}}^w$. This contradicts the fact that $t_{ij}^w \leq t_{i\overline{j}}^w$ that follows from $\overline{j} \in J_y$. □

Following Lu (2013), the robustness cost of solution $y \in \Omega$ corresponds to the maximal possible robust deviation if sites J_y are located:

$$RC(y) = \max_{w \in \mathcal{W}} DEV(w, y) \tag{5.3}$$

We denote by *worst-case scenario* a scenario which solves (5.3). The $(RPCP_2)$ aims to minimize the regret in the worst-case scenario for all feasible solution $y \in \Omega$:

$$(RPCP_2) : \min_{y \in \Omega} RC(y) \tag{5.4}$$

We now show that it is not necessary to consider the whole uncertainty set W to optimally solve $(RPCP_2)$.

5.3.2 Reducing the number of scenarios

Since a box-uncertainty set contains an infinite number of scenarios for a given solution $y \in \Omega$, the evaluation of the robustness cost (5.3) is a major challenge when solving $(RPCP_2)$. We prove that it is sufficient to consider N scenarios per solution $y \in \Omega$ to optimally solve $(RPCP_2)$, i.e., $N \cdot \binom{M}{p}$ scenarios, instead of the $p^N \cdot N \cdot \binom{M}{p}$ scenarios considered in Lu (2013) for $(RPCP_1)$.

Definition 5.3.1. *Let $y \in \Omega$ be a feasible solution and let $\bar{i} \in \mathcal{N}$ be any demand node. We define $w_{\bar{i}}(y)$ as the following scenario:*

- $d_i^{\omega_{\bar{i}}(y)} = \begin{cases} d_i^+ & \text{if } i = \bar{i} \\ d_i^- & \text{otherwise} \end{cases}$
- $t_{ij}^{\omega_{\bar{i}}(y)} = \begin{cases} t_{ij}^+ & \text{if } i = \bar{i} \text{ and } y_j = 1 \\ t_{ij}^- & \text{otherwise} \end{cases}$

Hence, in scenario $w_{\bar{i}}(y)$, node $\bar{i}$ is at its maximal demand value while the other nodes are at their minimal demand value. Also, the traveling time of any node-site pair is set to its minimal value except for node $\bar{i}$ for which they are maximal. We prove in the following that at least one of the scenarios in $\{w_i(y)\}_{i\in\mathcal{N}}$ leads to a maximal deviation for $y \in \Omega$.

Theorem 5.3.1. *Let $y \in \Omega$ be a first-stage solution of $(RPCP_2)$. There exists $i \in \mathcal{N}$ such that $w_i(y)$ is an optimal solution for $RC(y)$.*

Proof: Let $w^0 \in \mathcal{W}$ be an optimal solution to $RC(y)$ (i.e., $RC(y) = DEV(w^0, y)$). We built a sequence of three other optimal scenarios w^1, w^2, w^3 and show that w^3 is equal to $w_i(y)$ for some $i \in \mathcal{N}$.

Let us consider, for any $k \in \{0, 1, 2, 3\}$, the (node, site) pair $(i_k, j_k) \in \mathcal{N} \times J_y$ such that $Z(w^k, y) = d_{i_k}^{w^k} t_{i_k j_k}^{w^k}$ (i.e., (i_k, j_k) allow to reach the optimal radius $Z(w^k, y)$).

<u>*Scenario w^1*</u>

Let w^1 be a scenario identical to w^0 except for $d_{i_0}^{w^1}$ which is equal to $d_{i_0}^+$. This amounts to multiplying any weighted travel time from i_0 by the same constant $\frac{d_{i_0}^+}{d_{i_0}^{w^0}}$. This ensures that an optimal radius in w^1 is still obtained for (i_0, j_0) and that one can set $(i_1, j_1) = (i_0, j_0)$. We demonstrate below that the deviation of w^1 is not lower than that of w^0.

$$DEV(w^1, y) - DEV(w^0, y) = Z(w^1, y) - Z(w^0, y) - \Big[Z(w^1, y^*(w^1)) - Z(w^0, y^*(w^0))\Big] \quad (5.5)$$

It holds that:

$$Z(w^1, y) - Z(w^0, y) = d_{i_1}^{w^1} t_{i_1 j_1}^{w^1} - d_{i_0}^{w^0} t_{i_0 j_0}^{w^0} = d_{i_0}^+ t_{i_0 j_0}^{w^0} - d_{i_0}^{w^0} t_{i_0 j_0}^{w^0} = (d_{i_0}^+ - d_{i_0}^{w^0}) t_{i_0 j_0}^{w^0} \quad (5.6)$$

Now, let $\tilde{j} = \underset{j \in J_{y^*(w^0)}}{\arg\min}\, t^{w^0}_{i_0 j}$, i.e., node i_0 is optimally assigned to site $\tilde{j}$ when sites set $J_{y^*(w^0)}$ of solution $y^*(w^0)$ are opened. The aim is to prove the following inequality:

$$Z(w^1, y^*(w^1)) - Z(w^0, y^*(w^0)) \leq (d^+_{i_0} - d^{w^0}_{i_0})\ t^{w^0}_{i_0 \tilde{j}} \tag{5.7}$$

First, $Z(w^1, y^*(w^1)) \leq Z(w^1, y^*(w^0))$ holds from the definition of $y^*(w)$. Second, let us consider solution $y^*(w^0)$ and its opened sites set $J_{y^*(w^0)}$. In scenario w^0, node i_0 is assigned to site $\tilde{j}$ at distance $d^{w^0}_{i_0} t^{w^0}_{i_0 \tilde{j}}$. In scenario w^1, node i_0 is optimally assigned to the same site $\tilde{j}$, at distance $d^+_{i_0} t^{w^0}_{i_0 \tilde{j}}$. All the other nodes are assigned in the same way in w^0 and in w^1. Therefore, Inequality (5.7) is satisfied.

From (5.5), (5.6), and (5.7) we deduce the following inequality:

$$DEV(w^1, y) - DEV(w^0, y) \geq (d^+_{i_0} - d^{w^0}_{i_0})(t^{w^0}_{i_0 j_0} - t^{w^0}_{i_0 \tilde{j}}) \tag{5.8}$$

It remains to prove that $t^{w^0}_{i_0 j_0} - t^{w^0}_{i_0 \tilde{j}} \geq 0$. It comes from the definition of $\tilde{j}$ and Z that $d^{w^0}_{i_0} t^{w^0}_{i_0 \tilde{j}} \leq Z(w^0, y^*(w^0))$. It also comes from the definition of $y^*(w)$ that $Z(w^0, y^*(w^0)) \leq Z(w^0, y) = d^{w^0}_{i_0} t^{w^0}_{i_0 j_0}$. Consequently, $DEV(w^1, y)$ is at least as large as $DEV(w^0, y)$, and scenario w^1 is optimal for $RC(y)$.

Scenario w^2

Let w^2 be a scenario identical to w^1 except for $t^{w^1}_{i_1 j}$ which is equal to $t^+_{i_1 j}$ for all $j \in J_y$. This ensures that an optimal radius in w^2 is still obtained for (i_1, j_1) and that one can set $(i_2, j_2) = (i_1, j_1)$, since i_1 was already leading to the largest weighted travel time in w^1 and some of its travel times have been increased in w^2. Let $\bar{j} = \underset{j \in J_{y^*(w^1)}}{\arg\min}\, t^{w^1}_{i_1 j}$, i.e., node i_1 is optimally assigned to site $\bar{j}$ when sites set $J_{y^*(w^1)}$ of solution $y^*(w^1)$ are opened. Two cases are discussed:

- Case 1: $\bar{j} \in J_y$, i.e., site $\bar{j}$ is open in y. In this case, we can deduce from Lemma 5.3.1 that $DEV(w^1, y) = 0$. As w^1 is an optimal scenario for $RC(y)$, we deduce that $DEV(w^2, y) \leq 0$ and, as deviations are non-negative, $DEV(w^2, y) = 0$. Scenario w^2 is also optimal.
- Case 2: $\bar{j} \notin J_y$. Here, solution $y^*(w^1)$ is also optimal for scenario w^2 since node i_1 is assigned to $\bar{j} \notin J_y$ when sites $J_{y^*(w^1)}$ are located, and since the only difference between w^1 and w^2 is an increase of the travel time between i_1 and the sites in J_y. Consequently, $Z(w^1, y^*(w^1)) = Z(w^2, y^*(w^2))$. Moreover, the increase of travel times leads to $Z(w^2, y) \geq Z(w^1, y)$ which ensures that $DEV(w^2, y) \geq DEV(w^1, y)$. Therefore, scenario w^2 is optimal for $RC(y)$.

Scenario w^3

Let w^3 be a scenario identical to w^2 except for $d_i^{w^3}$ which is equal to d_i^- for all $i \in \mathcal{N}\backslash\{i_2\}$ and $t_{ij}^{w^3}$ which is equal to t_{ij}^- for all $i \in \mathcal{N}\backslash\{i_2\}$ and $j \in \mathcal{M}$. This ensures that it is possible to have $(i_3, j_3) = (i_2, j_2)$ since the demand and travel times of i_2 are not modified and the others are reduced. This also ensures that $Z(w^3, y) = Z(w^2, y)$ and that $Z(w^3, y^*(w^3))$ is not greater than $Z(w^2, y^*(w^2))$. Consequently, $DEV(w^3, y)$ is not lower than $DEV(w^2, y)$ which proves that $w_{i_3}(y)$ is an optimal scenario.

Finally, since $i_3 = i_2 = i_1 = i_0$, it is possible to sum up the changes that were progressively made on w^0 to reach w^3, and observe that w^3 is precisely $w_{i_0}(y)$. Therefore, scenario $w_{i_0}(y)$ is optimal for $RC(y)$. □

Since Ω is finite, Theorem 5.3.1 enables to only consider a finite set of scenarios $\overline{\mathcal{W}} = \{w_i(y) \mid y \in \Omega,\ i \in \mathcal{N}\}$ without losing the optimality:

$$(RPCP_2) : \min_{y\in\Omega} \{ \max_{w\in\overline{\mathcal{W}}} DEV(w, y)\} \tag{5.9}$$

5.3.3 MILP formulations of the robust weighted vertex p-center problem

We present how the five formulations presented in Section (5.2.1) can be adapted to solve $(RPCP_2)$. Let Z_w^* be the optimal value of the (PCP) problem when the uncertain parameters take value $w \in \overline{\mathcal{W}}$ (i.e., $Z_w^* = Z(w, y^*(w))$). Note that in the following formulations, the value of Z_w^* is assumed to be known for all $w \in \overline{\mathcal{W}}$. Section 5.3.6 presents how these values can be computed efficiently in the proposed algorithm when required.

The robust formulation based on $(F1)$ uses one set of allocation variables x_{ij}^w for each scenario $w \in \overline{\mathcal{W}}$ to allow different demand node allocations depending on the scenario:

$$(RF1)\begin{cases}
\min & RC & & (5.10)\\
\text{s.t.} & RC \geq \sum_{j\in\mathcal{M}} d_i^w t_{ij}^w x_{ij}^w - Z_w^* & i \in \mathcal{N},\ w \in \overline{\mathcal{W}} & (5.11)\\
& \sum_{j\in\mathcal{M}} x_{ij}^w = 1 & i \in \mathcal{N},\ w \in \overline{\mathcal{W}} & (5.12)\\
& x_{ij}^w \leq y_j & i \in \mathcal{N},\ j \in \mathcal{M},\ w \in \overline{\mathcal{W}} & (5.13)\\
& \sum_{j\in\mathcal{M}} y_j = p & & \\
& y_j \in \{0,1\} & j \in \mathcal{M} & \\
& x_{ij}^w \in \{0,1\} & i \in \mathcal{N},\ j \in \mathcal{M},\ w \in \overline{\mathcal{W}} & (5.14)
\end{cases}$$

Constraints (5.11) set a lower bound on the value of the robustness cost (RC) for each scenario. Objective (5.10) provides a solution with the lowest maximal deviation. This formulation contains an exponential number of variables and constraints as the size of $\overline{W}$ is proportional to $|\Omega|$.

Now, to adapt the other formulations to $(RPCP_2)$, let K^w be the number of different values $\{d_i^w t_{ij}^w\}_{i\in\mathcal{N},j\in\mathcal{M}}$ and let $\mathcal{K}^w$ be the set of indices $\{1,\dots,K^w\}$ for each scenario $w \in \overline{W}$. We need also to sort these values in increasing order and obtain a set of distinct distances D_w^k for $k \in \mathcal{K}^w$. We can also identify the following sub-set of indices for each scenario $w \in \mathcal{W}$ and each client $i \in \mathcal{N}$:

$$\mathcal{K}_i^w = \{k \in \mathcal{K}^w : \exists\, j \in \mathcal{M} \text{ such that } d_{ij} = D_w^k\}$$

Consequently, regarding formulation $(F2)$, considering a set of variables of radius z_w^k, we obtain:

$$(RF2)\begin{cases} \min & RC & & & \\ \text{s.t.} & RC \geq D_w^1 + \sum_{k=2}^{K^w}\left(D_w^k - D_w^{k-1}\right)z_w^k - Z_w^* & w \in \overline{\mathcal{W}} & (5.15)\\ & z_w^k + \sum_{j:d_i^w t_{ij}^w < D_w^k} y_j \geq 1 & w \in \overline{\mathcal{W}},\ i \in \mathcal{N},\ k \in \mathcal{K}_i^w & (5.16)\\ & z_w^k \geq z_w^{k+1} & w \in \overline{\mathcal{W}},\ k \in \mathcal{K}^w \setminus \{K^w\} & (5.17)\\ & \sum_{j\in\mathcal{M}} y_j = p & & \\ & y_j \in \{0,1\} & j \in \mathcal{M} & \\ & z_w^k \in \{0,1\} & w \in \overline{\mathcal{W}},\ k \in \mathcal{K}^w & (5.18) \end{cases}$$

Formulation $(F3)$ does not directly provide the value R of the optimal radius but its index r instead, such that $D^r = R$. This raises a problem when considering the adaptation of $(F3)$ to the solution of $(RPCP_2)$ as a given index does not necessarily correspond to the same distance in different scenarios. Consequently, $(F3)$ is first modified so that it provides a distance rather than its index. Constraints (4.23) are replaced by:

$$R \geq D^k\Big(1 - \sum_{j:d_i t_{ij} < D^k} y_j\Big) \qquad \forall\, i \in \mathcal{N}, k \in \mathcal{K}_i \qquad (5.19)$$

We can now obtain a reformulation of $(RPCP_2)$ based on $(F3)$:

$$(RF3)\begin{cases} \min & RC & & \\ \text{s.t.} & RC \geq D_w^k(1 - \sum_{j:d_i^w t_{ij}^w < D_w^k} y_j) - Z_w^* & w \in \overline{\mathcal{W}},\ i \in \mathcal{N},\ k \in \mathcal{K}_i^w & (5.20) \\ & \sum_{j \in \mathcal{M}} y_j = p & & \\ & y_j \in \{0,1\} & j \in \mathcal{M} & \end{cases}$$

Similarly to $(RF2)$, with the same set of distinct distances D_w^k, $(F4)$ can be adapted to $(RPCP_2)$.

$$(RF4)\begin{cases} \min & RC & & \\ \text{s.t.} & RC \geq \sum_{k \in \mathcal{K}^w} D_w^k u_w^k - Z_w^* & w \in \overline{\mathcal{W}} & (5.21) \\ & \sum_{j:d_i^w t_{ij}^w \leq D_w^k} y_j \geq \sum_{q=1}^{k} u_w^q & w \in \overline{\mathcal{W}},\ i \in \mathcal{N},\ k \in \mathcal{K}^w & (5.22) \\ & \sum_{k \in \mathcal{K}^w} u_w^k = 1 & w \in \overline{\mathcal{W}} & (5.23) \\ & \sum_{j \in \mathcal{M}} y_j = p & & \\ & y_j \in \{0,1\} & j \in \mathcal{M} & \\ & u_w^k \in \{0,1\} & w \in \overline{\mathcal{W}},\ k \in \mathcal{K}^w & \end{cases}$$

Finally, $(F5)$ can also be directly adapted to $(RPCP_2)$.

$$(RF5)\begin{cases} \min & RC & & \\ \text{s.t.} & RC \geq d_i^w t_{ij}^w - \sum_{j':d_i^w t_{ij'}^w < d_i^w t_{ij}^w} (d_i^w t_{ij}^w - d_i^w t_{ij'}^w) y_{j'} - Z_w^*, & w \in \overline{\mathcal{W}},\ i \in \mathcal{N},\ j \in \mathcal{M} & (5.24) \\ & \sum_{j \in \mathcal{M}} y_j = p & & \\ & y_j \in \{0,1\} & j \in \mathcal{M} & \\ & RC \geq 0 & & \end{cases}$$

Note that both $(RF3)$ and $(RF5)$ do not require an exponential number of variables, which is a significant advantage compared to $(RF1)$, $(RF2)$, and $(RF4)$. However, the five reformulations have an exponential number of constraints.

5.3.4 Column-and-constraint generation algorithm

To solve these robust MILP formulations, we propose the *column-and-constraint generation* algorithm ($C\&CG$) represented in Algorithm 12 in which ($\overline{RF}$) can be any of the proposed five robust formulations ($RF1$), ($RF2$), ($RF3$), ($RF4$), or ($RF5$) with $\overline{\mathcal{W}}$ initially empty. At each iteration, we generate a solution (y, RC) which satisfies all the scenarios currently in $\overline{W}$ by solving ($\overline{RF}$) (Step 3). If the solution does not satisfy one of the scenarios $\{w_i(y)\}_{i\in\mathcal{N}}$ (Step 10), the most violated scenario is added to $\overline{W}$ (Step 14). When no violated scenario is found, an optimal solution is returned. The value of the optimal radius considering a scenario $w_i(y)$ can be calculated by solving a deterministic (PCP) (Step 7). Note that the radius associated with a feasible solution y in a scenario w_i can be obtained quickly as it only requires to determine the distance between each demand node and its closest center in J_y (Step 8).

Algorithm 12: *column-and-constraint generation* algorithm

input :

- Instance data ($\mathcal{N}$, $\mathcal{M}$, p, $[d_i^-, d_i^+]$ and $[t_{ij}^-, t_{ij}^+]$ for each $i \in \mathcal{N}$ and $j \in \mathcal{M}$).
- A robust formulation ($\overline{RF}$) for the (PCP).

output:

- An optimal solution y of ($\overline{RF}$) and its robustness cost RC.

1 $RC \leftarrow 0$, $\overline{\mathcal{W}} \leftarrow \emptyset$, $isOptimal \leftarrow false$
2 **while** $isOptimal = false$ **do**
3 $(y, RC) \leftarrow$ solve ($\overline{RF}$) with scenarios $\overline{W}$
4 $isOptimal \leftarrow true$
5 $\overline{w} \leftarrow \emptyset$
6 **for** $i \in \mathcal{N}$ **do**
7 $Z^* \leftarrow$ optimal radius of the deterministic (PCP) for scenario $w_i(y)$
8 $Z \leftarrow$ radius for y in scenario $w_i(y)$
9 $DEV \leftarrow Z - Z^*$
10 **if** $DEV > RC$ **then**
11 $isOptimal \leftarrow false$
12 $RC \leftarrow DEV$
13 $\overline{w} \leftarrow w_i(y)$
14 $\overline{\mathcal{W}} \leftarrow \overline{\mathcal{W}} \cup \{\overline{w}\}$
15 **return** y and RC

5.3.5 Branch-and-cut algorithm

The main advantage of $(RF3)$, and $(RF5)$ over $(RF1)$, $(RF2)$, and $(RF4)$ is that no new variable is required when a scenario is added to $\overline{W}$. Consequently, using $(RF3)$ or $(RF5)$, we can define a *branch-and-cut* algorithm $(B\&C)$ which checks, at each node of the search tree, if each obtained integer solution (y, RC) satisfies all the scenarios $\{w_i(y)\}_{i\in\mathcal{N}}$. If it does not, the corresponding violated inequalities are generated and the solution is ignored by the solver. This can be performed through callbacks which is a feature provided by mixed integer programming solvers. Consequently, Steps 4 to 14 of Algorithm 12 are performed within a callback. This modification allows us to only generate a single search tree instead of solving $(\overline{RF})$ from scratch at each iteration of the while loop.

5.3.6 Solving the deterministic p-center problems

One of the most time consuming steps in these two algorithms is solving the deterministic (PCP) associated with the current feasible solution $y \in \Omega$ and scenario $w_i(y)$ in order to obtain $Z^*_{w_i(y)}$ for each $i \in \mathcal{N}$ (Step 7 in Algorithm 12). Any of the formulations presented in Section 5.2.1 could be considered to solve these deterministic problems. However, to improve the performances, the two following improvements are considered.

Reducing the number of deterministic problems solved

For a given $i \in \mathcal{N}$, if it is known in advance that a solution y satisfies scenario $w_i(y)$ (i.e., that $Z(w_i(y), y) - Z^*_{w_i(y)} \leq RC$), the solution of the associated deterministic (PCP) can be avoided. In particular, this is the case if it is known a lower bound Z^{*i}_{lb} on $Z^*_{w_i(y)}$ such that $Z(w_i(y), y) - Z^{*i}_{lb} \leq RC$. Indeed, in that case $Z(w_i(y), y) - Z^*_{w_i(y)} \leq Z(w_i(y), y) - Z^{*i}_{lb} \leq RC$ holds. Now, let us consider the scenarios defined as follows.

Definition 5.3.2. *Let $\bar{i} \in \mathcal{N}$. We define $w^{\bar{i}}_{lb}$ as the following scenario:*

- $d_i^{w^{\bar{i}}_{lb}} = \begin{cases} d_i^+ & \text{if } i = \bar{i} \\ d_i^- & \text{otherwise} \end{cases}$
- $t_{ij}^{w^{\bar{i}}_{lb}} = t_{ij}^- \qquad \forall j \in \mathcal{M}$

The following lemma shows that the optimal value of the deterministic (PCP) associated to w^i_{lb} provides a lower bound on $Z^*_{w_i(y)}$.

Lemma 5.3.2. *For any $i \in \mathcal{N}$ and $y \in \Omega$, $Z^*_{w^i_{lb}} \leq Z^*_{w_i(y)}$.*

Proof:

It is known that $Z^*_{w^i_{lb}} \leq Z(w^i_{lb}, y^*(w_i(y)))$. Moreover, since the only difference between scenarios $w_i(y)$ and w^i_{lb} is a reduction of travel times $Z(w^i_{lb}, y^*(w_i(y))) \leq Z^*_{w_i(y)}$. □

Note that scenarios $\{w^i_{lb}\}_{i\in\mathcal{N}}$ do not depend on any feasible solution $y \in \Omega$ and therefore the N corresponding lower bounds can all be computed in a pre-processing step. This improvement enables a significant reduction of the solution time.

Binary search

Solving deterministic (PCP) through MILP solvers is not the most efficient approach. As mentioned in the previous Chapter 4, the best methods are those of Contardo et al. (2019) and Gaar and Sinnl (2022). The former initially considers a subset of clients, solves the (PCP) associated to this subset, and adds new clients until an optimal solution is obtained. The latter method is based on a branch-and-cut algorithm, considering Benders cuts strengthened with a lifting procedure, and a specialized separation scheme. Both methods can solve very large-scale instances. However, these methods are sophisticated enough to do our own implementation. In order to reach a good compromise between performance and ease of implementation, the deterministic (PCP) is solved through a binary search algorithm well known in the literature (Toregas et al. (1971)) which we have already presented in the Section 4.6.2.

5.3.7 Adaptation to the single-stage problem

In the single-stage problem $(RPCP_1)$, both the location of centers and the demand node allocations are fixed before the uncertainty is revealed. Consequently, allocation variables x are necessary to compute the radius. Given a feasible solution (x, y), let t^w_i be the travel time in scenario $w \in \mathcal{W}$ between a demand node $i \in \mathcal{N}$ and its allocated center (i.e., $t^w_i = t^w_{ij}$ with $j \in \mathcal{M}$ the only center such that $x_{ij} = 1$). Thus, the radius of solution (x, y) is $\max_{i\in\mathcal{N}} d^w_i t^w_i$ and $(RPCP_1)$ can be defined as:

$$(RPCP_1): \min_{(x,y)} \max_{w\in\mathcal{W}} \left\{ \max_{i\in\mathcal{N}} d^w_i t^w_i - Z^*_w \right\} \tag{5.25}$$

where Z^*_w is the optimal solution of the deterministic p-center problem in which the uncertain parameters take value w.

The $(C\&CG)$ and $(B\&C)$ algorithms previously presented can be adapted to solve the $(RPCP_1)$ using $(RF1)$. For this purpose, one single set of allocation variables x_{ij} must be considered

which ensures that the demand node allocations are the same regardless of the scenario. These adaptations are not possible for the $(C\&CG)$ and $(B\&C)$ algorithms based on the others formulations. Indeed, in these formulations it is not possible to ensure that the demand node allocations are the same in all scenarios. Considering only one set of variables z_k in $(RF2)$ or u_k in $(RF4)$ would only ensure that the distance index of the radius is the same in each scenario, not that the demand node allocations are. For the algorithms based on $(RF3)$ or $(RF5)$ the adaptation seems even less possible as these formulation do not contain scenario variables and as the demand node allocations are determined implicitly in the corresponding constrains. Note that for $(RPCP_1)$ the finite set of scenarios that a solution must satisfy to ensure its optimality is different from $\{w_i(y)\}_{i\in\mathcal{N}}$ as proved in Theorem 1 of Lu (2013).

5.4 Computational study

To evaluate the efficiency of the proposed $(C\&CG)$ and $(B\&C)$ algorithms, a case study presented in Lu (2013) and randomly generated instances are studied. It was not possible to make a direct comparison with Lu (2013), because the solution values presented in Lu (2013) are not consistent with the ones obtained by the proposed exact solution approach. This is detailed in Section 5.5.

This study was carried out on an Intel(R) Xeon(R) Gold 6144 processor 3.5 GHz, with 32 threads, but only 1 was used, and 378 GB of RAM. IBM ILOG CPLEX 20.1 was used as solver. For the $(B\&C)$ algorithm, the GenericCallback of CPLEX is used, which gets called whenever a feasible integer solution is found. The absolute tolerance to the best integer objective (EpGap) is set to 10^{-10}. All times presented in the tables are CPU times in seconds. All theses instances files are available online[1].

5.4.1 Case Study

Lu (2013) presents a case study on the location of urgent relief distribution centers (URDCs) in a relief supply distribution network responding to the massive earthquake which hit central Taiwan on September 21, 1999. Specifically, relief supplies were collected from six unaffected counties transported to two URDCs, and then delivered to the 51 relief stations in the 11 townships. Five other candidate sites for URDCs were considered. They divided the number of survivors by the number of relief stations of each township to estimate the relief demand faced by each relief station. They use the data collected in previous research for the travel time between a URDC and a relief station.

[1] https://osf.io/87u6f/

Following Lu (2013), we consider a distance uncertainty box $[t_{ij},\ t_{ij}(1+\alpha_1)]$, and the demand uncertainty box $[d_i(1-\alpha_2),\ d_i(1+\alpha_2)]$ with $\alpha_1 \in \{0.5,\ 1.5,\ 2.5\}$ and $\alpha_2 \in \{0.2,\ 0.4,\ 0.6\}$. Thus, we consider the 9 combinations of parameter values α_1 and α_2 considering 51 demand nodes, 7 possible sites, and the selection of 2 centers. Table 5.2 shows the results obtained when solving $(RPCP_2)$ by applying the proposed algorithms to the case study.

Instance						Iterations							Time [s]						
						C&CG					B&C		C&CG					B&C	
N	M	p	α_1	α_2	RC	RF1	RF2	RF3	RF4	RF5	RF3	RF5	RF1	RF2	RF3	RF4	RF5	RF3	RF5
51	7	2	0.5	0.2	495,000	5	5	5	5	5	5	5	1.0	1.0	**0.9**	10.0	**0.9**	1.2	1.2
51	7	2	0.5	0.4	838,500	6	6	6	6	6	10	6	1.1	1.0	1.0	14.1	**0.9**	1.2	1.1
51	7	2	0.5	0.6	1,159,200	7	6	6	6	6	6	6	1.3	1.0	**0.8**	11.7	0.9	1.1	1.1
51	7	2	1.5	0.2	1,238,400	14	16	16	15	16	16	15	5.7	4.3	**1.9**	153.5	**1.9**	**1.9**	**1.9**
51	7	2	1.5	0.4	1,705,800	16	15	15	14	16	16	15	8.0	3.0	**1.8**	109.1	**1.8**	1.9	**1.8**
51	7	2	1.5	0.6	2,150,400	16	15	15	14	15	14	15	7.2	2.8	1.7	100.3	**1.6**	1.8	1.8
51	7	2	2.5	0.2	1,981,800	18	19	22	19	21	18	18	11.6	5.4	2.5	238.9	2.5	**2.2**	**2.2**
51	7	2	2.5	0.4	2,573,100	19	19	21	19	20	18	19	12.4	4.7	2.3	209.9	**2.2**	**2.2**	**2.2**
51	7	2	2.5	0.6	3,141,600	20	19	19	19	19	19	19	13.4	4.3	**2.1**	181.7	**2.1**	**2.1**	**2.1**
Average						13	13	14	13	14	14	13	6.9	3.0	**1.6**	114.3	**1.6**	1.7	1.7

Table 5.2: Results of $C\&CG$ and $B\&C$ algorithms for the $(RPCP_2)$ on the case study instances.

Note that the optimal value of $(RPCP_2)$ increases with α_1 and α_2 in Table 5.2. The fastest algorithm is $(C\&CG)$ using the $(RF3)$ and $(RF5)$ formulations, closely followed by the two $(B\&C)$. Unlike the deterministic problem, $(RF2)$ and $(RF4)$ are not the most efficient. Moreover, $(RF4)$ requires many more variables and constraints than the other formulations, resulting in a much longer solving time per iteration of the $(C\&CG)$ algorithm.

We observe that the robust solution of all these instances also is the deterministic solution which consists of opening sites 3 and 4. This may be due to the fact that the bounds of the uncertainty boxes are all increased in the same proportion for all the nodes. To avoid this, random intervals for all uncertain parameters are considered in the following.

5.4.2 Randomly generated instances

Following Lu (2013), two dimensional coordinates were uniformly drawn from $[0,\ 100] \times [40,\ 60]$ for the two set of demand nodes and available sites. The distances t_{ij} between demand nodes and sites were set to the nearest integer of their Euclidean distance. The demand d_i of each demand node $i \in \mathcal{N}$ was uniformly drawn from the interval $[1000,\ 2000]$. Nevertheless, we do not consider the same uncertainty sets as in Lu (2013). For all $i \in \mathcal{N}$ and $j \in \mathcal{M}$, t_{ij} and d_i can take any value in $[t_{ij},\ t_{ij}(1+\alpha_1^{ij})]$ and $[d_i(1-\alpha_2^i),\ d_i(1+\alpha_2^i)]$. Three cases are considered depending on whether α_1^{ij} and α_2^i are randomly generated in $[0.1,\ 0.3]$, $[0.4,\ 0.6]$, or $[0.7,\ 0.9]$.

Firstly, 18 instances are generated for $(RPCP_2)$ considering $N = M \in \{15, 25, 40\}$, and $p = \{2, 3\}$. The results obtained for these instances are presented in Tables 5.3, and 5.4 respectively. The last columns represent the robust solution and the deterministic solutions. Since both problems may have several optimal solutions, we include in the tables the solutions that are the most frequent among those of the seven methods considered. For theses instances, a time-limit of 600 seconds (TL2) is considered.

Similarly to Table 5.2, both RC and the solving time increase with α_1 and α_2 in this first set of instances. $(RF3)$ and $(RF5)$ are also the fastest $(C\&CG)$ algorithms. This could be explained by the fact that the addition of a scenario does not lead to the addition of any variable. The $(B\&C)$ algorithms are much faster even though they perform more iterations. The better performances of $(RF5)$ with the $(B\&C)$ algorithm is due to its smaller number of iterations.

We observe that the box uncertainty sets with random bounds lead to robust solutions which are different from the deterministic one, in contrast to the case study. Note that the CPU time increases with p. Indeed, none of the $(C\&CG)$ algorithms is able to solve all the instances for $p = 3$. This could be explained by the number of feasible solutions which is proportional to $\binom{M}{p}$.

Instance					Iterations							Time [s]							Robust Solution		Nominal Solution	
					C&CG					B&C		C&CG					B&C					
N	M	p	(α_1, α_2)	RC	RF1	RF2	RF3	RF4	RF5	RF3	RF5	RF1	RF2	RF3	RF4	RF5	RF3	RF5				
15	15	2	[0.1, 0.3]	45,870	5	8	6	5	5	8	8	0.5	1.4	**0.4**	8.7	**0.4**	**0.4**	0.5	10	14		
15	15	2	[0.4, 0.6]	136,587	17	16	15	14	17	21	19	6.1	3.2	0.6	31.1	0.7	0.5	**0.5**	11	14	11	14
15	15	2	[0.7, 0.9]	228,429	34	28	33	33	33	37	35	26.4	3.4	1.3	144.5	2.0	1.6	**1.1**	11	14		
25	25	2	[0.1, 0.3]	40,205	8	8	8	8	8	16	12	3.1	6.6	**1.0**	184.9	**1.0**	1.2	1.1	21	22		
25	25	2	[0.4, 0.6]	142,096	21	20	21	16	22	32	30	41.6	33.3	3.0	TL	3.7	2.1	**1.7**	7	22	13	21
25	25	2	[0.7, 0.9]	228,195	36	34	38	20	38	47	35	230.9	14.5	4.5	TL	5.3	2.2	**1.9**	13	15		
40	40	2	[0.1, 0.3]	54,082	12	12	13	3	11	19	14	51.4	58.8	**4.1**	TL	7.4	8.0	6.9	10	26		
40	40	2	[0.4, 0.6]	143,310	23	23	34	3	33	45	36	TL	TL	94.5	TL	63.0	18.0	**14.7**	25	26	8	26
40	40	2	[0.7, 0.9]	283,119	30	56	89	6	96	122	108	TL	TL	67.6	TL	91.2	17.6	**15.4**	23	32		
			Average				29		29	39	33			19.7		19.4	5.7	**4.9**				

Table 5.3: Results of $C\&CG$ and $B\&C$ algorithms on randomly generated $(RPCP_2)$ instances with $N = M \in \{15, 25, 40\}$ and $p = 2$. TL2=600s.

Instance					Iterations							Time [s]							Robust Solution			Nominal Solution		
					C&CG					B&C		C&CG					B&C							
N	M	p	(α_1, α_2)	RC	RF1	RF2	RF3	RF4	RF5	RF3	RF5	RF1	RF2	RF3	RF4	RF5	RF3	RF5						
15	15	3	[0.1, 0.3]	56,689	8	7	8	8	8	10	11	0.7	0.7	**0.4**	15.5	0.5	**0.4**	**0.4**	1	6	8			
15	15	3	[0.4, 0.6]	97,360	16	15	16	15	17	24	16	3.6	5.1	0.8	59.7	1.0	0.7	**0.5**	1	6	12	1	6	12
15	15	3	[0.7, 0.9]	166,252	33	36	35	37	36	44	43	19,6	14.4	1.5	200.3	2.0	**1.0**	**1.0**	1	6	12			
25	25	3	[0.1, 0.3]	40,950	12	8	7	7	10	16	8	6.7	4.7	0.9	157.3	1.7	1.1	**0.7**	2	12	14			
25	25	3	[0.4, 0.6]	84,827	16	16	18	13	18	45	21	18.6	27.3	6.3	TL	5.5	3.2	**2.0**	2	14	50	3	7	25
25	25	3	[0.7, 0.9]	165,978	50	51	54	20	57	66	64	305.8	94.8	14.6	TL	17.9	**3.9**	4.1	6	7	17			
40	40	3	[0.1, 0.3]	48,123	18	17	15	3	18	40	33	213.1	TL	20.6	TL	22.7	9.4	**8.5**	25	26	40			
40	40	3	[0.4, 0.6]	125,024	23	19	54	4	52	197	103	TL	TL	389.7	TL	161.7	26.9	**18.1**	18	25	35	8	26	40
40	40	3	[0.7, 0.9]	236,529	33	44	152	6	132	537	589	TL	TL	TL	TL	TL	**71.2**	74.9	25	29	40			
			Average							109	99						13.1	**12.2**						

Table 5.4: Results of $C\&CG$ and $B\&C$ algorithms on randomly generated $(RPCP_2)$ instances with $N = M \in \{15, 25, 40\}$ and $p = 3$. TL2=600s.

Since the limit of the $(C\&CG)$ algorithms is reached, the following will focus on the $(B\&C)$ algorithms. We now consider a time limit of 9000 seconds and instances considering $N = M \in \{60, 80, 100\}$, and $p = \{2, 3, 4, 5\}$. The results are presented in Tables 5.5, 5.6, 5.7, and 5.8.

Both algorithms were able to solve most instances. However, due to memory limitations, the optimal solution is not obtained for all instances with 80 and 100 nodes in particular for those with the largest uncertainty sets. The results of the two formulations are similar, with $(RF3)$ performing better on average for $p = \{3, 4, 5\}$, and worse for $p = 2$. Their solving time is still closely related to the number of iterations even if for a few instances, $(RF3)$ performs more iterations but is faster.

Instance					B&C				Robust Solution				Nominal Solution	
					Iterations		Time [s]							
N	M	p	(α_1, α_2)	RC	RF3	RF5	RF3	RF5	RF3		RF5			
60	60	2	[0.1, 0.3]	45,075	31	15	16	**9**	9	32	9	32		
60	60	2	[0.4, 0.6]	112,704	81	38	37	**21**	31	32	31	32	4	32
60	60	2	[0.7, 0.9]	216,104	176	73	66	**37**	32	45	32	45		
80	80	2	[0.1, 0.3]	37,245	21	18	**25**	26	2	61	2	61		
80	80	2	[0.4, 0.6]	147,825	117	62	113	**74**	27	61	27	61	54	61
80	80	2	[0.7, 0.9]	243,722	338	296	**251**	275	25	27	25	27		
100	100	2	[0.1, 0.3]	53,119	55	26	109	**66**	33	65	65	84		
100	100	2	[0.4, 0.6]	146,640	183	154	265	**241**	9	64	33	64	33	64
100	100	2	[0.7, 0.9]	246,597	460	497	**630**	716	64	84	64	84		
Average					162	131	168	**163**						

Table 5.5: Results of $B\&C$ algorithm on randomly generated $(RPCP_2)$ instances with $N = M \in \{60, 80, 100\}$ and $p = 2$.

Instance					B&C				Robust Solution						Nominal Solution		
					Iterations		Time [s]										
N	M	p	(α_1, α_2)	RC	RF3	RF5	RF3	RF5	RF3			RF5					
60	60	3	[0.1, 0.3]	45,430	92	79	29	**27**	24	32	36	24	32	56			
60	60	3	[0.4, 0.6]	117,659	354	310	**127**	**127**	10	42	44	10	33	38	2	32	48
60	60	3	[0.7, 0.9]	209,352	1015	1024	**444**	507	20	31	42	14	31	42			
80	80	3	[0.1, 0.3]	33,439	55	48	**83**	95	32	41	58	32	41	58			
80	80	3	[0.4, 0.6]	128,900	386	332	399	**336**	32	63	74	32	63	74	9	71	73
80	80	3	[0.7, 0.9]	206,001	980	1033	**1295**	1647	10	58	61	10	58	61			
100	100	3	[0.1, 0.3]	35,780	52	58	**122**	175	12	76	84	12	76	84	12	32	83
100	100	3	[0.4, 0.6]	124,704	498	323	1403	**1192**	19	88	89	12	19	88			
Average					429	401	**488**	513									

Table 5.6: Results of $B\&C$ algorithm on randomly generated $(RPCP_2)$ instances with $N = M \in \{60, 80, 100\}$ and $p = 3$.

Instance					B&C				Robust Solution								Nominal Solution			
					Iterations		Time [s]		RF3				RF5							
N	M	p	(α_1, α_2)	RC	RF3	RF5	RF3	RF5												
60	60	4	[0.1, 0.3]	36,737	50	42	44	**36**	3	8	18	55	2	8	18	55				
60	60	4	[0.4, 0.6]	99,544	326	293	**379**	408	2	8	26	32	2	26	32	35	2	8	32	43
60	60	4	[0.7, 0.9]	171,749	1684	1519	**2918**	3371	26	32	48	53	12	26	48	53				
80	80	4	[0.1, 0.3]	31,788	92	58	482	**325**	3	32	58	69	21	32	58	60	1	3	32	58
80	80	4	[0.4. 0.6]	108,312	736	633	**2152**	3259	1	32	48	55	17	32	39	55				
100	100	4	[0.1, 0.3]	32,841	203	142	**941**	1816	12	14	35	44	12	14	35	73	1	4	12	35
Average					515	448	**1153**	1536												

Table 5.7: Results of $B\&C$ algorithm on randomly generated $(RPCP_2)$ instances with $N = M \in \{60, 80, 100\}$ and $p = 4$.

Instance					B&C				Robust Solution									Nominal Solution				
					Iterations		Time [s]		RF3					RF5								
N	M	p	(α_1, α_2)	RC / BKV	RF3	RF5	RF3	RF5														
60	60	5	[0.1, 0.3]	29,709	52	28	51	**31**	2	8	12	18	37	3	8	18	18	37					
60	60	5	[0.4, 0.6]	86,466	386	185	970	**774**	2	8	18	28	38	2	8	28	28	38	2	8	12	18	40
60	60	5	[0.7, 0.9]	138,852	888	1323	**5596**	6461	2	8	21	38	39	2	21	38	38	39					
80	80	5	[0.1, 0.3]	29,259	147	129	3086	**2366**	1	44	48	58	59	1	44	58	58	59	11	15	16	18	40
80	80	5	[0.4, 0.6]	99,449	643	547	TL3	TL3	2	11	42	47	73	2	11	47	31	73					
100	100	5	[0.1, 0.3]	31,107	142	133	**803**	6066	13	14	90	95	100	13	14	95	95	100	3	12	14	50	85
Average					372	395	**2011**	2747														

Table 5.8: Results of $B\&C$ algorithm on randomly generated $(RPCP_2)$ instances with $N = M \in \{60, 80, 100\}$ and $p = 5$. TL3=9000s.

5.4.3 ORLIB instances

We consider the deterministic ORLIB instances (Beasley (1990)), which are symmetrical instances, since the set of clients is also the set of candidate sites. In these instances the travel times t_{ij} are provided. To create $(RPCP2)$ instances, demand values in $[1, 100]$ were randomly generated and for the travel times the same box uncertainty set as in the previous section were considered, with α_i and α_{ij} randomly generated in $[0.1, 0.9]$ and such that $\alpha_{ij} = \alpha_{ji}$ for $i \in \mathcal{N}$ and $j \in \mathcal{M}$. The results are presented in Tables 5.9, 5.10, and 5.11. For theses instances, a time-limit of 9000 seconds (TL3) was considered.

Starting from $p = 2$, we observe that the solving time increases rapidly with p. Particularly, instance pmed3 is only solved for $p = 2$ and instance pmed1 could not be solved for $p = 4$ due to memory issues. As for the random instances, the formulations have similar performances which depend strongly on the number of iterations. It was not possible to solve larger ORLIB instances with $N \geq 200$ due to memory limitations.

Instance					B&C				Robust Solution				Nominal Solution	
Name	$N = M$	p	(α_1, α_2)	RC	Iterations RF3	Iterations RF5	Time [s] RF3	Time [s] RF5	RF3		RF5			
pmed1	100	2	[0.1, 0.9]	14,706	392	494	**511**	663	8	57	8	57	60	95
pmed2	100	2	[0.1, 0.9]	16,138	595	626	**672**	738	23	64	23	64	7	21
pmed3	100	2	[0.1, 0.9]	25,443	717	633	825	**758**	15	88	15	88	89	99
pmed4	100	2	[0.1, 0.9]	19,293	761	778	**1070**	1134	32	100	32	99	2	98
pmed5	100	2	[0.1, 0.9]	13,336	267	186	274	**193**	52	68	52	68	15	52
Average					546	543	**670**	697						

Table 5.9: Results of *B&C* algorithm on adapted ORLIB instances for the ($RPCP_2$) with $N = M = 100$ and $p = 2$.

Instance					B&C				Robust Solution						Nominal Solution		
Name	$N = M$	p	(α_1, α_2)	RC	Iterations RF3	Iterations RF5	Time [s] RF3	Time [s] RF5	RF3			RF5					
pmed1	100	3	[0.1, 0.9]	14,094	675	587	2876	**2188**	9	42	94	9	42	64	9	13	94
pmed2	100	3	[0.1, 0.9]	17,718	567	615	**1137**	1245	35	65	88	35	68	88	2	9	22
pmed4	100	3	[0.1, 0.9]	15,423	395	459	**1303**	1413	32	88	99	32	51	99	1	69	87
pmed5	100	3	[0.1, 0.9]	12,228	601	536	1348	**1346**	51	68	97	51	68	97	7	32	46
Average					560	549	1666	**1548**									

Table 5.10: Results of *B&C* algorithm on adapted ORLIB instances for the ($RPCP_2$) with $N = M = 100$ and $p = 3$.

Instance					B&C				Robust Solution								Nominal Solution			
name	$N = M$	p	(α_1, α_2)	RC	Iterations RF3	Iterations RF5	Time [s] RF3	Time [s] RF5	RF3				RF5							
pmed2	100	4	[0.1, 0.9]	12,600	1032	1134	**3457**	4980	23	58	77	97	23	58	76	97	2	8	22	40
pmed4	100	4	[0.1, 0.9]	13,812	507	648	6299	**4463**	7	26	52	99	7	26	52	99	8	23	51	55
pmed5	100	4	[0.1, 0.9]	11,650	1150	-	**5563**	-	60	71	90	97	-	-	-	-	23	51	69	97

Table 5.11: Results of *B&C* algorithm on adapted ORLIB instances for the ($RPCP_2$) with $N = M = 100$ and $p = 4$. TL3=9000s.

5.5 Comparison of results with Lu (2013)

It was not possible to compare the performances of proposed exact algorithms and the one of the heuristic in Lu (2013) as its results do not seem to be correct. We prove that several robustness costs obtained with this heuristic and reported in Lu (2013) are undervalued.

In ($RPCP_1$), the demand node allocations are fixed before the uncertainty is revealed. As defined in Section 5.3.7, given a feasible solution (x, y), let t_i^w be the distance (or travel time) in scenario $w \in \mathcal{W}$ between a demand node $i \in \mathcal{N}$ and its allocated center (i.e., $t_i^w = t_{ij}^w$ with $j \in \mathcal{M}$ the only center such that $x_{ij} = 1$). Thus, the radius of solution (x, y) is $\max_{i \in \mathcal{N}} d_i^w t_i^w$ and its robustness cost is $RC(x, y) = \max_{w \in \mathcal{W}} \left\{ \max_{i \in \mathcal{N}} d_i^w t_i^w - Z_w^* \right\}$, where Z_w^* is the optimal solution of the deterministic p-center problem in which the uncertain parameters take value w.

A difficulty to evaluate the robustness costs reported in Lu (2013) is that for each solution only one demand node allocation is provided. Let us consider the instance described in Section 5.4.1 in which $p = 2$, $\alpha_1 = 0.5$, and $\alpha_2 = 0.2$ and let (x^h, y^h) be its associated solution in Lu (2013). The only information available on (x^h, y^h) is that centers 1 and 2 are located ($x_1^h = x_2^h = 1$) and that demand node 21 is allocated to center 1 ($x_{21,1}^h = 1$). Nevertheless, this is sufficient to obtain a lower bound on the robustness cost as for any scenario $w \in \mathcal{W}$ and any demand node $i \in \mathcal{N}$, the expression $d_i^w t_i^w - Z_w^*$ constitutes a lower bound of $RC(x^h, y^h)$. Let us consider a scenario w_{21} in which the demand of node 21 is $d_{21}^w = d_{21}^+ = 34,800$ and its distance to center 1 is $t_{21,1}^w = t_{21,1}^+ = 41$. The optimal radius $Z_{w_{21}}^* = 495,600$ is obtained by solving a deterministic p-center problem. Consequently, a lower bound on the robustness cost of value $34,800 \times 41 - 495,600 = 931,200$ is obtained, which is higher than the value $93,619$ reported in Lu (2013).

Table 5.12 present similar results on nine instances. The third column contains the robustness costs reported in Lu (2013) which are almost all undervalued. Indeed, they are significantly lower than their associated lower bounds presented in Column 4. Column 5 contains the robustness cost of optimal solutions obtained by the proposed ($B\&C$) algorithm adapted to ($RPCP_1$) (see Section 5.3.7). Note that the branch-and-cut always returns a solution which robustness cost is always lower than the lower bound of the heuristic solution.

Instance		Robustness cost		
		Heuristic solution from Lu (2013)		Optimal solution
α_1	α_2	Results from Lu (2013)	According to this article	
0.5	0.2	= 93,619	≥ 931,200	= 495,000
0.5	0.4	= 587,837	≥ 1,292,900	= 838,500
0.5	0.6	= 1,477,709	≥ 1,906,600	= 1,159,200
1.5	0.2	= 940,858	≥ 1,870,800	= 1,238,400
1.5	0.4	= 1,859,069	≥ 2,389,100	= 1,705,800
1.5	0.6	= 2,934,605	≥ 3,362,600	= 2,150,400
2.5	0.2	= 1,883,309	≥ 2,810,400	= 1,981,800
2.5	0.4	= 3400,291	≥ 4,029,900	= 2,573,100
2.5	0.6	= 4,356,480	≥ 4,129,600	= 3,141,600

Table 5.12: Comparison of the robustness cost of solutions obtained by the heuristic presented in Lu (2013) and optimal solutions obtained by the branch-and-cut algorithm adapted for $(RPCP_1)$.

5.6 Conclusions

The solution of a robust weighted vertex p-center problem is studied, considering uncertain nodal weights demand and edge lengths using box uncertainty sets. Two variants of this problem are possible depending on whether the demand node allocations to the centers are made after the uncertainty is revealed $(RPCP_2)$ or not $(RPCP_1)$.

We prove that for $(RPCP_2)$ a finite subset of scenarios from the box uncertainty set can be considered without losing optimality. This result is used to propose five robust reformulations based on different MILP formulations of the vertex p-center problem. To optimally solve these reformulations, a *column-and-constraint generation* algorithm and a *branch-and-cut* algorithm are introduced. Moreover, we identify a lower bound on the optimal value of the deterministic p-center problem associated with the finite subset of scenarios. This result is used to significantly reduce the solving time of proposed algorithms. Finally, we highlight how proposed methods can be adapted to optimally solve $(RPCP_1)$.

We present a numerical study to compare the performances of the algorithms on a case study, on randomly generated instances, and on a few instances from ORLIB. We were able to solve optimally the 68 considered instances. The *column-and-constraint generation* algorithm based on formulation $(RF3)$ and $(RF5)$ are more efficient than the one based on $(RF1)$, $(RF2)$, and $(RF4)$. This is because adding a scenario does not require the addition of any variable. This formulation enables the implementation of a *branch-and-cut* algorithm which significantly reduces the solving time.

In future work, analysis of larger instances with other random box uncertainty sets could be considered. To further improve the performances of the *branch-and-cut* algorithm, alternative branching strategies could be evaluated and integrality cuts (UserCuts) could be dynamically generated. The algorithms could also be improved by solving the deterministic (PCP) at each iteration with one of the state-of-the-art exact methods of Contardo et al. (2019) or Gaar and Sinnl (2022).

Chapter 6

Conclusion

This chapter presents some discussion on the main results of this thesis. Furthermore, some perspectives that motivate future work and new research directions are presented.

6.1 Conclusions

Discrete location problems have been well studied in the literature. Nevertheless, there are still many studies that seek to improve or propose new solution methods. Furthermore, although there is a great development of approximation methods, there are still major challenges for the exact solution of large-scale problems. We have studied the p-center problem (PCP) and p-median problem (PMP) which are fundamental problems in location science and therefore have many applications in operations research. In these problems, the number p of installations to be located is known in advance, and opening costs are not considered. We have focused on how their corresponding MILP formulations impact their solution methods.

We have studied Benders decomposition for both p-center and p-median problems, which, along with efficient cut separation algorithms and other enhancements, constitutes the current state of the art in exact solution methods for large-scale instances. Specifically, for (PMP) we have been able to solve different types of instances with more than 238,000 sites, and clients. Improving the results of the best approach presented in García et al. (2011). For (PCP) we also studied the Benders decomposition and found a class of new Benders cuts. These cuts can be implemented as a generalization for the already known ones. Although our implementation cannot outperform the work of Gaar and Sinnl (2022) in terms of execution time, our method gives good results in terms of linear relaxation bounds. On the other hand, our proposed algorithm based on the clustering

procedure can outperform the state-of-the-art methods of Gaar and Sinnl (2022) and Contardo et al. (2019) for large values of p. We have been able to solve instances for more than 744,000 sites and clients. The good performance of our algorithm is based on several improvements such as the two-stage implementation, the efficient cluster update which identifies good lower and upper bounds, and an exact iterative distance rounding procedure.

We have also studied the incorporation of uncertainty into the (PCP) problem in order to study some realistic problems. We were the first to consider uncertainty in distances and client demands simultaneously for a robust two-stage problem. We have considered uncertainty intervals for each parameter, for which we prove that it is sufficient to consider a finite subset of scenarios without losing optimality. Our research has produced five new robust reformulations that minimize the maximal regret in relation to the worst-case scenario. Furthermore, we propose the use of lower bounds on the worst-case scenario that allow our exact algorithms to solve instances efficiently.

6.2 Perspectives

Regarding the (PMP), we have considered only separating cuts for integer solutions in the *branch-and-cut*, an implementation that also separate fractional solutions could be studied. In addition, there are other branching strategies that could be designed.

In relation to (PCP), we show that there are different approaches within the same principal resolution method. The in-depth study of the new cuts may bring new theoretical and implementation results. Moreover, the techniques of client clustering, dominated facilities, and distance rounding have allowed us to have the best resolution performance, therefore we think that each of these can be improved and used in other location problems.

We think that our work on the robust (PCP) can be the basis for other solution methods as well as for other variants of the (PCP), and other uncertainty sets, or other objective functions of the robust problem. Moreover, incorporating other techniques in our *branch-and-cut* should be able to further improve the performance to solve instances of a larger size.

Bibliography

Al-Khedhairi, A., S. (2005). Enhancements to two exact algorithms for solving the vertex P-center problem. *Journal of Mathematical Modelling and Algorithms Volume*, 4:129–147.

Ales, Z. and Elloumi, S. (2018). Compact milp formulations for the p-center problem. In *International Symposium on Combinatorial Optimization*, pages 14–25. Springer.

An, Y., Zeng, B., Zhang, Y., and Zhao, L. (2014). Reliable p-median facility location problem: two-stage robust models and algorithms. *Transportation Research Part B: Methodological*, 64:54–72.

Avella, P., Boccia, M., Salerno, S., and Vasilyev, I. (2012). An aggregation heuristic for large scale p-median problem. *Computers & Operations Research*, 39(7):1625–1632.

Avella, P., Sassano, A., and Vasil'ev, I. (2007). Computational study of large-scale p-median problems. *Mathematical Programming*, 109(1):89–114.

Averbakh, I. and Berman, O. (1997). Minimax regret p-center location on a network with demand uncertainty. *Location Science*, 5(4):247–254.

Averbakh, I. and Berman, O. (2000). Algorithms for the robust 1-center problem on a tree. *European Journal of Operational Research*, 123(2):292–302.

Baron, O., Milner, J., and Naseraldin, H. (2011). Facility location: A robust optimization approach. *Production and Operations Management*, 20.

Basu, S., Sharma, M., and Ghosh, P. S. (2015). Metaheuristic applications on discrete facility location problems: a survey. *OPSEARCH*, 52(3):530–561.

Beasley, J. E. (1990). Or-library: Distributing test problems by electronic mail. *The Journal of the Operational Research Society*, 41(11):1069–1072.

Ben-Tal, A., Ghaoui, L., and Nemirovski, A. (2009). *Robust Optimization*. Princeton Series in Applied Mathematics. Princeton University Press.

Benders, J. F. (1962). Partitioning procedures for solving mixed-variables programming problems. *Numerische Mathematik*, 4(1):238–252.

Bertsimas, D., Litvinov, E., Sun, X. A., Zhao, J., and Zheng, T. (2013). Adaptive robust optimization for the security constrained unit commitment problem. *IEEE Transactions on Power Systems*, 28(1):52–63.

Bertsimas, D. and Sim, M. (2004). The price of robustness. *Operations Research*, 52(1):35–53.

Briant, O. and Naddef, D. (2004). The optimal diversity management problem. *Operations research*, 52(4):515–526.

Çalık, H., Labbé, M., and Yaman, H. (2019a). *p-Center Problems*, pages 51–65. Springer International Publishing, Cham.

Çalık, H., Labbé, M., and Yaman, H. (2019b). p-center problems. In *Location science*, pages 51–65. Springer.

Calik, H. and Tansel, B. C. (2013). Double bound method for solving the p-center location problem. *Computers and Operations Research*, 40(12):2991–2999.

Caruso, C., Colorni, A., and Aloi, L. (2003). Dominant, an algorithm for the p-center problem. *European Journal of Operational Research*, 149(1):53–64.

Chan, T. C. Y., Shen, Z.-J. M., and Siddiq, A. (2018). Robust defibrillator deployment under cardiac arrest location uncertainty via row-and-column generation. *Operations Research*, 66(2):358–379.

Chen, D. and Chen, R. (2009). New relaxation-based algorithms for the optimal solution of the continuous and discrete p-center problems. *Computers and Operations Research*, 36(5):1646–1655.

Cheng, C., Adulyasak, Y., and Rousseau, L.-M. (2021). Robust facility location under demand uncertainty and facility disruptions. *Omega*, 103:102429.

Church, R. L. (1984). Symposium on location problems: In memory of leon cooper. *Journal of Regional Science*, 24(2):185–201.

Contardo, C., Iori, M., and Kramer, R. (2019). A scalable exact algorithm for the vertex p-center problem. *Computers and Operations Research*, 103:211–220.

Cordeau, J.-F., Furini, F., and Ljubić, I. (2019). Benders decomposition for very large scale partial set covering and maximal covering location problems. *European Journal of Operational Research*, 275(3):882 – 896.

Cornuejols, G., Nemhauser, G. L., and Wolsey, L. A. (1980). A canonical representation of simple plant location problems and its applications. *SIAM Journal on Algebraic Discrete Methods*, 1(3):261–272.

Daskin, M. (1996). Network and discrete location: Models, algorithms and applications. *Journal of the Operational Research Society*, 48.

Demange, M., Gabrel, V., Haddad, M. A., and Murat, C. (2020). A robust p-center problem under pressure to locate shelters in wildfire context. *EURO Journal on Computational Optimization*, 8:103–139.

Du, B. and Zhou, H. (2018). A robust optimization approach to the multiple allocation p-center facility location problem. *Symmetry*, 10(11):588.

Du, B., Zhou, H., and Leus, R. (2020). A two-stage robust model for a reliable p-center facility location problem. *Applied Mathematical Modelling*, 77:99–114.

Duran-Mateluna, C., Ales, Z., and Elloumi, S. (2023a). An efficient Benders decomposition for the p-median problem. *European Journal of Operational Research*, 308(1):84–96.

Duran-Mateluna, C., Ales, Z., Elloumi, S., and Jorquera-Bravo, N. (2023b). Robust MILP formulations for the two-stage weighted vertex p-center problem. *Computers & Operations Research*, 159:106334.

Elloumi, S. (2010). A tighter formulation of the p-median problem. *Journal of Combinatorial Optimization*, 19(1):69–83.

Elloumi, S., Labbé, M., and Pochet, Y. (2004). A New Formulation and Resolution Method for the p-Center Problem. *INFORMS Journal on Computing*, 16(1):84–94.

Elloumi, S. and Plateau, A. (2010). A computational study for the p-median problem. *Electronic Notes in Discrete Mathematics*, 36:455–462.

Fischetti, M., Ljubic, I., and Sinnl, M. (2017). Redesigning benders decomposition for large-scale facility location. *Management Science*, 63(7):2146–2162.

Gaar, E. and Sinnl, M. (2022). A scaleable projection-based branch-and-cut algorithm for the p-center problem. *European Journal of Operational Research*.

Galvão, R. D. (1980). A dual-bounded algorithm for the p-median problem. *Operations Research*, 28(5):1112–1121.

García, S., Labbé, M., and Marín, A. (2011). Solving large p-median problems with a radius formulation. *INFORMS Journal on Computing*, 23(4):546–556.

Garcia-Diaz, J., Menchaca-Mendez, R., Menchaca-Mendez, R., Pomares Hernández, S., Pérez-Sansalvador, J. C., and Lakouari, N. (2019). Approximation algorithms for the vertex k-center problem: Survey and experimental evaluation. *IEEE Access*, 7:109228–109245.

Garfinkel, R. S., Neebe, A. W., and Rao, M. R. (1977). The m-Center Problem: Minimax Facility Location. *Management Science*, 23(10):1133–1142.

Hakimi, S. (1964). Optimum locations of switching centers and the absolute centers and medians of a graph. *Operations Research*, 12:450–459.

Hakimi, S. L. (1965). Optimum distribution of switching centers in a communication network and some related graph theoretic problems. *Operations research*, 13(3):462–475.

Hansen, P., Brimberg, J., Urošević, D., and Mladenović, N. (2009). Solving large p-median clustering problems by primal–dual variable neighborhood search. *Data Mining and Knowledge Discovery*, 19(3):351–375.

Hasani, A. and Mokhtari, H. (2018). Redesign strategies of a comprehensive robust relief network for disaster management. *Socio-Economic Planning Sciences*, 64:92–102.

Irawan, C. and Salhi, S. (2015a). Aggregation and non aggregation techniques for large facility location problems: A survey. *Yugoslav Journal of Operations Research*, 25:1–1.

Irawan, C. A. and Salhi, S. (2015b). Solving large p-median problems by a multistage hybrid approach using demand points aggregation and variable neighbourhood search. *Journal of*

Global Optimization, 63(3):537–554.

Irawan, C. A., Salhi, S., and Scaparra, M. P. (2014). An adaptive multiphase approach for large unconditional and conditional p-median problems. *European Journal of Operational Research*, 237(2):590–605.

Kariv, O. and Hakimi, S. L. (1979). An algorithmic approach to network location problems. ii: The p-medians. *SIAM Journal on Applied Mathematics*, 37(3):539–560.

Klastorin, T. D. (1985). The p-Median Problem for Cluster Analysis: A Comparative Test Using the Mixture Model Approach. *Management Science*, 31(1):84–95.

Laporte, G., Nickel, S., and da Gama, F. (2019a). *Facility Location Under Uncertainty*, pages 185–213. Springer International Publishing, Cham.

Laporte, G., Nickel, S., and Saldanha-da Gama, F. (2019b). *Location Science (2nd ed)*. Springer International Publishing.

Lu, C.-C. (2013). Robust weighted vertex p-center model considering uncertain data: An application to emergency management. *European Journal of Operational Research*, 230(1):113–121.

Lu, C.-C. and Sheu, J.-B. (2013). Robust vertex p-center model for locating urgent relief distribution centers. *Computers & Operations Research*, 40(8):2128–2137.

Magnanti, T. L. and Wong, R. T. (1981). Accelerating benders decomposition: Algorithmic enhancement and model selection criteria. *Operations Research*, 29(3):464–484.

Marín and Pelegrín, M. (2019). *The p-Median Problem*, pages 25–50. Springer International Publishing.

Minieka, E. (1970). The m-center problem. *SIAM Review*, 12(1):138–139.

Mladenović, N., Brimberg, J., Hansen, P., and Moreno-Pérez, J. A. (2007). The p-median problem: A survey of metaheuristic approaches. *European Journal of Operational Research*.

Mu, W. and Tong, D. (2020). On solving large p-median problems. *Environment and Planning B: Urban Analytics and City Science*, 47(6):981–996.

Nikoofal, M. E. and Sadjadi, S. J. (2010). A robust optimization model for p-median problem with uncertain edge lengths. *The International Journal of Advanced Manufacturing Technology*, 50:391–397.

Park, H.-S. and Jun, C.-H. (2009). A simple and fast algorithm for K-medoids clustering. *Expert Systems with Applications*, 36(2, Part 2):3336–3341.

Paul, J. A. and Wang, X. J. (2015). Robust optimization for united states department of agriculture food aid bid allocations. *Transportation Research Part E: Logistics and Transportation Review*, 82:129–146.

Paul, J. A. and Wang, X. J. (2019). Robust location-allocation network design for earthquake

preparedness. *Transportation research part B: methodological*, 119:139–155.

R. Chen, G. H. (1987). Relaxation method for the solution of the minimax location-allocation problem in euclidean space. *av. Res. Logist.*, 34:775–788.

Rahmaniani, R., Crainic, T. G., Gendreau, M., and Rei, W. (2017). The benders decomposition algorithm: A literature review. *European Journal of Operational Research*, 259(3):801 – 817.

Reese, J. (2006). Solution methods for the p-median problem: An annotated bibliography. *Networks*, 48(3):125–142.

Reinelt, G. (1991). TSPLIB—A Traveling Salesman Problem Library. *ORSA Journal on Computing*, 3(4):376–384.

Resende, M. G. C. and Werneck, R. F. (2004). A Hybrid Heuristic for the p-Median Problem. *Journal of Heuristics*, 10(1):59–88.

ReVelle, C. S. and Eiselt, H. A. (2005). Location analysis: A synthesis and survey. *European Journal of Operational Research*, 165(1):1–19.

ReVelle, C. S. and Swain, R. W. (1970). Central facilities location. *Geographical Analysis*, 2(1):30–42.

Serra, D. and Marianov, V. (1998). The p-median problem in a changing network: the case of barcelona. *Location Science*, 6(1-4):383–394.

Sheu, J.-B. (2007). An emergency logistics distribution approach for quick response to urgent relief demand in disasters. *Transportation Research Part E: Logistics and Transportation Review*, 43(6):687–709.

Snyder, L. V. (2006). Facility location under uncertainty: a review. *IIE transactions*, 38(7):547–564.

T. Ilhan, F.A. Özsoy, M. P. (2002). An efficient exact algorithm for the vertex p-center problem and computational experiments for different set covering subproblems . *Technical Report*.

Takedomi, S., Ishigaki, T., Hanatsuka, Y., and Mori, T. (2022). Facility location optimization with pMP modeling incorporating waiting time prediction function for emergency road services. *Computers & Industrial Engineering*, 164:107859.

Toregas, C., Swain, R., ReVelle, C., and Bergman, L. (1971). The location of emergency service facilities. *Operations research*, 19(6):1363–1373.

Trivedi, A. and Singh, A. (2017). A hybrid multi-objective decision model for emergency shelter location-relocation projects using fuzzy analytic hierarchy process and goal programming approach. *International Journal of Project Management*, 35(5):827–840.

Trivedi, A. and Singh, A. (2019). Shelter planning for uncertain seismic hazards using multicriteria decision approach: a case of nepal earthquake. *Journal of Multi-Criteria Decision Analysis*, 26(3-4):99–111.

Ushakov, A. V. and Vasilyev, I. (2021). Near-optimal large-scale k-medoids clustering. *Information Sciences*, 545:344–362.

Voevodski, K. (2021). Large Scale K-Median Clustering for Stable Clustering Instances. In Banerjee, A. and Fukumizu, K., editors, *Proceedings of The 24th International Conference on Artificial Intelligence and Statistics*, volume 130 of *Proceedings of Machine Learning Research*, pages 2890–2898. PMLR.

Zeng, B. and Zhao, L. (2013). Solving two-stage robust optimization problems using a column-and-constraint generation method. *Operations Research Letters*, 41(5):457–461.

ECOLE
DOCTORALE
DE MATHEMATIQUES
HADAMARD

Titre : Méthodes de solutions exactes pour les problèmes de localisation discrète de p-installations à grande échelle.
Mots clés : Recherche opérationnelle, optimisation combinatoire, optimisation robuste, localisation discrète, p-centre, p-median.

Résumé : Cette thèse porte sur la solution exacte des problèmes NP-difficiles du p-median et du p-centre, des problèmes d'optimisation combinatoire qui deviennent rapidement difficiles à résoudre lorsque la taille de l'instance augmente. Ces problèmes de localisation discrète consistent à ouvrir un nombre défini p d'installations, puis à leur affecter un ensemble de clients selon une fonction objectif à minimiser.
Tout d'abord, nous étudions le problème du p-median qui cherche à minimiser la somme des distances entre les clients et les installations ouvertes auxquelles ils sont affectés. Nous développons un algorithme basé sur la décomposition de Benders qui surpasse les méthodes exactes de l'état de l'art. L'algorithme considère une approche en deux étapes et ainsi qu'un algorithme efficace pour la séparation des coupes de Benders. Cette méthode est évaluée sur plus de 230 instances de benchmark avec jusqu'à 238,025 clients et sites. De nombreuses instances sont résolues à l'optimalité pour la première fois ou ont leur meilleure solution connue améliorée.
Deuxièmement, nous explorons le problème du p-centre qui cherche à minimiser la plus grande distance entre un client et l'installation ouverte qui en est la plus proche. Nous comparons d'abord les cinq principales formulations MILP de la littérature. Nous étudions la décomposition de Benders et nous proposons également un algorithme exact basé sur une procédure de partionnement des clients reposant sur la structure du problème. Nous avons été en mesure de résoudre des instances avec jusqu'à 744,710 clients et sites. Toutes les méthodes proposées sont comparées à l'état de l'art dans des instances de benchmark. Les résultats obtenus sont analysés, mettant en évidence les avantages et les inconvénients de chaque méthode.
Enfin, nous étudions un problème robuste du p-centre en deux étapes avec une incertitude sur les demandes et les distances des nœuds. Nous introduisons la reformulation robuste du problème basée sur les cinq principales formulations déterministes MILP de la littérature. Nous prouvons que seul un sous-ensemble fini de scénarios de l'ensemble d'incertitude infini peut être pris en compte sans perdre l'optimalité. Nous proposons également un algorithme de génération de colonnes et de contraintes et ainsi qu'un algorithme de branch-and-cut pour résoudre efficacement ce problème. Nous montrons comment ces algorithmes peuvent également être adaptés pour résoudre le problème robuste d'une seule étape. Les différentes formulations proposées sont testées sur des instances générées aléatoirement et sur un cas d'étude de la littérature.

Title: Exact solution methods for large-scale discrete p-facility location problems.
Key words: Operations research, combinatorial optimisation, robust optimisation, discrete location, p-centre, p-median.

Abstract: This thesis focuses on the exact solution of the NP-hard problems p-median and p-center, combinatorial optimization problems that quickly become difficult to solve as the instance size increases. These discrete location problems involve opening a defined number p of facilities and then allocating to them a set of clients according to an objective function to be minimized.
Firstly, we study the p-median problem, which seeks to minimize the sum of distances between clients and the open facilities to which they are allocated. We develop an algorithm based on Benders decomposition that outperforms state-of-the-art exact methods. The algorithm considers a two-stage approach and an efficient algorithm for separating Benders cuts. The method has been evaluated on over 230 benchmark instances with up to 238,025 clients and sites. Many instances are solved to optimality for the first time or have their best known solution improved.
Secondly, we explore the p-center problem, which seeks to minimize the largest distance between a client and its nearest open facility. We first compare the five main MILP formulations in the literature. We study the Benders decomposition and also propose an exact algorithm based on a client clustering procedure based on the structure of the problem. All the proposed methods are compared with the state-of-the-art on benchmark instances. We have been able to solve instances with up to 744,710 clients and sites. The results obtained are analyzed, highlighting the advantages and disadvantages of each method.
Finally, we study a robust two-stage p-center problem with uncertainty on node demands and distances. We introduce the robust reformulation of the problem based on the five main deterministic MILP formulations in the literature. We prove that only a finite subset of scenarios from the infinite uncertainty set can be considered without losing optimality. We also propose a column and constraint generation algorithm and a branch-and-cut algorithm to efficiently solve this problem. We show how these algorithms can also be adapted to solve the robust single-stage problem. The different proposed formulations are tested on randomly generated instances and on a case study drawn from the literature.

Institut Polytechnique de Paris

91120 Palaiseau, France