UNIVERSITÉ DE MONTRÉAL

LE PROBLÈME PÉRIODIQUE DE TOURNÉES SUR LES ARCS AVEC CONTRAINTES DE CAPACITÉ ET DE GESTION DE STOCKS

JUAN PABLO RIQUELME RODRÍGUEZ
DÉPARTEMENT DE MATHÉMATIQUES ET DE GÉNIE INDUSTRIEL
ÉCOLE POLYTECHNIQUE DE MONTRÉAL

THÈSE PRÉSENTÉE EN VUE DE L'OBTENTION
DU DIPLÔME DE PHILOSOPHIÆ DOCTOR
(GÉNIE INDUSTRIEL)
JUIN 2014

UNIVERSITÉ DE MONTRÉAL

ÉCOLE POLYTECHNIQUE DE MONTRÉAL

Cette thèse intitulée :

LE PROBLÈME PÉRIODIQUE DE TOURNÉES SUR LES ARCS AVEC CONTRAINTES DE CAPACITÉ ET DE GESTION DE STOCKS

présentée par : RIQUELME RODRÍGUEZ Juan Pablo
en vue de l'obtention du diplôme de : Philosophiæ Doctor
a été dûment acceptée par le jury d'examen constitué de :

M. GENDREAU Michel, Ph.D., président
M. GAMACHE Michel, Ph.D., membre et directeur de recherche
M. LANGEVIN André, Ph.D., membre et codirecteur de recherche
M. ROUSSEAU Louis-Martin, Ph.D., membre
M. RENAUD Jacques, Ph.D., membre

DÉDICACE

À mes parents, Maria Dolores et Héctor,
ma sœur Elisa, et mes amis.

REMERCIEMENTS

Je tiens à remercier mes directeurs de recherche, les professeurs Michel Gamache et André Langevin, pour leurs observations, leur disponibilité corrections et leur orientation tout au long de ces quatre années. Mes remerciements aussi à tous les membres du jury pour avoir accepté de lire ce travail.

Je tiens à remercier au Conseil National de Science et Technologie (CONACYT) du Gouvernment du Mexique et au Groupe d'Études et de Recherche en Analyse de Désisions (GERAD).

Mes remerciements s'adressent, enfin, à ma soeur Elisa, mes parents Héctor et María Dolores, mes amis Trish, Marcela, Olivier, Aurora, Caro, Lalo, Tere et Francisco pour leur appui. Un spécial merci à Elspeth Adams pour son support et encouragement.

RÉSUMÉ

Dans cette thèse, on introduit le problème périodique de tournées sur les arcs avec contraintes de capacité et de gestion de stocks. Les arêtes d'un réseau représentent les clients qui nécessitent une certaine quantité de matériel. Ce matériel est mis en inventaire et consommé au cours du temps. Les besoins de réapprovisionnement indiquent la nature périodique du problème. Les exemples d'applications de ce problème sont l'arrosage des chemins de terre dans les mines à ciel ouvert pour supprimer la poussière, l'arrosage des routes dans les réseaux forestiers et l'arrosage des plantes sur les trottoirs des rues. On prend l'application de l'arrosage des routes dans les mines à ciel ouvert. Un camion-citerne se déplace le long des routes en arrosant de l'eau pour supprimer la poussière. À cause de sa capacité limitée, le camion doit retourner au dépôt avant de commencer une nouvelle tournée. À cause de l'évaporation de l'eau, l'humidité sur les routes diminue en fonction du temps. Les routes ont besoin d'un certain niveau d'humidité pour retenir efficacement les particules de poussière. Une pénurie arrive lorsque le niveau d'humidité se trouve en dessous du niveau requis. L'objectif de cette étude est de trouver un ensemble de tournées qui débutent et finissent au dépôt de telle façon que les coûts de pénalité liés à la pénurie, ainsi que les coûts de routage soient minimisés. Parce que l'ordre dans lequel les arêtes sont traversées et arrosées affecte le moment où l'humidité est restaurée, des décisions sur le routage et la gestion de l'inventaire sont prises simultanément. Ce problème a été traité pour les tournées sur les nœuds, i.e., les clients sont situés aux nœuds du réseau, et il est appelé *Inventory Routing Problem*. Cependant, il n'a pas été traité dans le domaine de tournées sur les arcs. Étant donné la capacité limitée du camion et la nature périodique du remplissage, on considère cette application comme un problème périodique de tournées sur les arcs avec contraintes de capacité (PCARP).

Au début, on considère le cas du problème d'arrosage où il n'existe qu'un seul dépôt (réservoir d'eau) dans le réseau et un seul camion citerne. On travaille sur un réseau mixte dans lequel, pour chaque arête, il y a deux arcs, un dans chaque direction de traverse. Il y a aussi une boucle artificielle au dépôt qui représente le remplissage du camion. L'horizon de temps est divisé en périodes de temps de même durée. Les coûts et les quantités en inventaire sont calculés pour chaque période de temps. On élabore un modèle de programmation linéaire en nombres entiers qui est testé pour des exemplaires connus du problème de tournées sur les arcs avec contraintes de capacité (CARP). La solution indique la séquence optimale de traverse et d'arrosage des arêtes, le remplissage du camion au dépôt, s'il a lieu, et les coûts totaux de routage et de pénalité pour la pénurie sur le niveau d'humidité. Les limites de ce

modèle sont établies en fonction de la taille des réseaux et de la longueur de l'horizon de temps qu'on est capable de résoudre. On est capable de trouver la solution optimale pour des réseaux avec 40 à 55 arêtes pour 20 à 30 périodes de temps. Ce qui correspond à un horizon de temps de 30 minutes en réalité. Deux situations sont testées, lorsque la quantité d'eau arrosée aux arêtes est variable ou constante. Les résultats sont présentés pour valider les deux situations. La contribution de cette première approche est le modèle mathématique pour résoudre le problème d'arrosage des routes dans les mines à ciel ouvert.

La deuxième approche a pour objectif de résoudre des exemplaires de plus grande taille et pour un horizon de temps plus long. On modifie le modèle mathématique pour inclure plus d'un véhicule et un seul dépôt. Avec ces modifications on est capable de trouver la solution optimale pour un exemplaire de petite taille, 11 arêtes, pour un horizon de temps de 20 minutes. Pour résoudre des exemplaires de plus grande taille et incrémenter l'horizon de temps, on utilise un algorithme heuristique appelé *adaptive large neighborhood search* (ALNS). L'ALNS se compose de huit opérateurs de destruction et de réparation choisis au hasard pour modifier la solution existante à chaque itération. La performance des opérateurs détermine la probabilité d'être choisi aux itérations suivantes. Une meilleure performance de l'opérateur, en termes d'amélioration de la solution existante, correspond à une plus grande probabilité d'être choisi. On utilise un ensemble d'exemplaires du CARP et un ensemble d'exemplaires créé à partir des réseaux de mines à ciel ouvert réels. Cette heuristique est capable de trouver une solution réalisable pour un horizon de temps de 300 minutes. Les opérateurs sont testés individuellement et en les combinant entre eux en utilisant un critère d'arrêt de 25000 itérations. On trouve la combinaison qui obtient la meilleure amélioration du coût total pour chaque ensemble d'exemplaires. Les contributions de cette approche sont la modification du modèle mathématique afin d'inclure plus d'un véhicule et l'application de l'heuristique ALNS pour obtenir une solution à ce nouveau problème.

Finalement, un dernier problème est abordé. Il consiste à localiser un ou plusieurs dépôts (réservoirs d'eau) le long des nœuds du réseau pour réduire les coûts de pénurie et de routage du problème d'arrosage des routes dans les mines à ciel ouvert. Comme l'activité principale se trouve sur les arêtes du réseau, ce problème correspond à un problème de localisation et de tournées sur les arcs (LARP) avec une composante périodique. Ce problème a été traité pour les tournées sur les nœuds. Cependant, il n'y a pas une autre application dans laquelle la localisation des dépôts est faite dans le domaine des problèmes périodiques de tournées sur les arcs. On prend des décisions à long terme telles que la localisation des dépôts et des décisions à court terme telles que le routage et la gestion des stocks. Pour cette raison, plusieurs

scénarios sont testés et leur coût moyen est ajouté aux coûts de localisation des dépôts afin d'obtenir un coût total pour le problème. Les scénarios sont le résultat de changements dans les paramètres du problème qui peuvent se produire sur un horizon de planification à long terme. Trois algorithmes de localisation sont utilisés pour obtenir une solution initiale à la localisation d'un et de plusieurs dépôts. Ces algorithmes suivent le processus *Location, allocation and Routing* (L-A-R), une méthode divisée en trois parties : premièrement, on place les dépôts sur les nœuds du réseau, puis on affecte les arêtes aux camions et finalement on trouve une tournée. L'heuristique ALNS développée pour l'approche précédente est adaptée et utilisée pour améliorer la solution. On compare la localisation d'un dépôt à différents endroits. On compare aussi les trois algorithmes de localisation. La contribution de cette partie est le développement d'un algorithme appliqué à la localisation de dépôts pour un problème périodique de tournées sur les arcs avec contraintes de capacité.

ABSTRACT

This dissertation introduces the periodic capacitated arc routing problem with inventory constraints. The edges of a network act as customers that require a certain quantity of material. It is then held as inventory and consumed over time. The need for replenishment of the consumed material explains the periodic nature of the problem. Some examples of applications of this problem are the road watering in open-pit mine roads to suppress dust, road watering in forest roads and plant watering on street medians and sidewalks. This work focuses on the application of road watering in open-pit mines. A water truck travels along the roads of a mine spraying water to suppress dust. Because of its limited capacity, the truck needs to replenish at a water depot before starting a new route. Due to water evaporation, the humidity on the roads decreases over time. Roads require a certain amount of humidity to effectively retain dust particles. A shortage happens when the humidity level drops below the required level. The objective of this thesis is to find a set of routes that start and end at the depot so that the penalty costs associated with shortage, as well as the routing costs are minimized. Because the order in which roads are traversed and watered affects their humidity level, routing and inventory decisions are made simultaneously. This problem has been treated for node routing, i.e., the customers are located at the nodes of the network, and it is called the Inventory Routing Problem. However, it has not being addressed in the arc routing domain. This problem is modeled as a periodic capacitated arc routing problem due to capacity constraints and the frequency of service.

The first case studied is where there is only one water depot and one vehicle to travel along the network. A mathematical model is developed using a mixed network. For each edge, there are two arcs that correspond to the direction in which the edge can be traversed. There is an artificial loop at the depot that represents the refill of the truck. The time horizon is divided in time periods of equal duration. Costs and inventory levels are calculated for each time period. The model is tested for known instances of the capacitated arc routing problem (CARP). It is able to solve to optimality networks of 40 to 55 edges for a time horizon of 20 to 30 periods. Two situations are considered where the quantity of water delivered to the edges is variable and constant. Results are reported to validate both situations. The contribution of this first approach is the mathematical model to solve the road watering problem.

The mathematical model is then modified to include more than one vehicle. As the number of variables increases, it is capable of solving to optimality a network of 11 edges for a time

horizon of less than 30 time periods. An adaptive large neighborhood search (ALNS) heuristic is developed to solve larger networks for a longer time horizon. It is able to provide a feasible solution for networks up to 55 edges and a time horizon of 300 time periods. The ALNS consists of an initial solution obtained using a construction algorithm and eight destroy-repair operators that are randomly selected to modify the initial solution at each iteration of the algorithm. The performance of these operators determines the probability of being selected for the next iteration. A better performance of the operator, in terms of improving the existing solution, corresponds to a higher probability of being selected. The operators are tested individually and in different combinations. The best combination is selected for each set of instances. Apart from the CARP instances, ten instances are created to test the algorithm. These new instances correspond to road networks of real open-pit mines. The contributions of this approach are the modification of the mathematical model to include more than one vehicle and the application of the ALNS to obtain a solution for this new problem.

Finally, a new problem is addressed. It consists in the location of one or more water depots along the nodes of the network to reduce the shortage and routing costs. Because the solution is obtained by servicing the edges of a network, this problem corresponds to a location arc routing problem (LARP) with a periodic component. This problem has only been treated in the node routing domain. No other application has been studied for location in the arc routing domain. Long term decisions, such as depot location, are combined with short term decisions, such as routing and inventory replenishment. Several scenarios are tested and their average cost is added to the depot placement costs in order to obtain a total cost. These scenarios are the result of changes in the parameters of the problem that can occur over a long planning horizon. Three location algorithms are used to obtain an initial solution to the location of one and several depots. The algorithms follow a location, allocation and routing (L-A-R) approach in which, first the depots are placed, then the edges are assigned to the service trucks and finally, a route is formed. The ALNS developed for the previous approach is adapted and used to improve the solution. The contribution is an algorithm applied to the location of depots for a periodic capacitated arc routing problem.

TABLE DES MATIÈRES

DÉDICACE iii

REMERCIEMENTS iv

RÉSUMÉ v

ABSTRACT viii

TABLE DES MATIÈRES x

LISTE DES TABLEAUX xiii

LISTE DES FIGURES xv

LISTE DES SIGLES ET ABRÉVIATIONS xvi

CHAPITRE 1 INTRODUCTION 1
1.1 Décisions de routage et gestion de stocks 2

CHAPITRE 2 REVUE DE LITTÉRATURE 7
2.1 Problèmes de tournées sur les arcs avec contraintes de capacité 7
2.1.1 Classement de problèmes de CARP 8
2.1.2 Méthodes de résolution du CARP 11
2.1.3 Métaheuristiques pour le CARP 15
2.2 CARP avec contraintes de périodicité (PCARP) 19
2.2.1 Modèles de programmation linéaire pour le PCARP 21
2.2.2 Heuristiques et métaheuristiques pour le PCARP 27
2.2.3 Autres applications du PCARP 31
2.3 Discussion 31

CHAPITRE 3 ARTICLE 1 : PERIODIC CAPACITATED ARC ROUTING PROBLEM WITH INVENTORY CONSTRAINTS 34
3.1 Abstract 36
3.2 Introduction 36
3.3 Literature review 37
3.4 Problem definition 38

3.5 Mixed integer programming model . . . 39
3.5.1 Inventory cost . . . 40
3.5.2 Routing cost . . . 42
3.5.3 Mathematical model . . . 42
3.5.4 Fixed-rate spraying . . . 46
3.5.5 Different vehicle speeds . . . 46
3.6 Test results . . . 47
3.6.1 Limits on the number of periods . . . 48
3.6.2 Comparison of models with fixed-rate and variable-rate spraying . . . 49
3.6.3 Computational time . . . 56
3.7 Conclusion and future work . . . 58

CHAPITRE 4 ARTICLE 2 : ADAPTIVE LARGE NEIGHBORHOOD SEARCH FOR THE PERIODIC CAPACITATED ARC ROUTING PROBLEM WITH INVENTORY CONSTRAINTS . . . 61
4.1 Abstract . . . 63
4.2 Introduction . . . 63
4.3 Problem definition . . . 64
4.3.1 The problem of watering roads in open-pit mines . . . 64
4.3.2 Mathematical model . . . 65
4.4 Adaptive large neighborhood search . . . 70
4.4.1 Initial solution . . . 71
4.4.2 Improvement phase . . . 77
4.4.3 The problem of inventory constraints . . . 79
4.5 Computational results . . . 79
4.5.1 Individual performance of the operators . . . 80
4.5.2 Parameter tuning . . . 83
4.5.3 Combinations of operators . . . 84
4.5.4 Networks with low percentage of improvement . . . 85
4.6 Conclusions and future work . . . 88
4.7 Acknowledgements . . . 88

CHAPITRE 5 ARTICLE 3 : LOCATION ARC ROUTING PROBLEM WITH INVENTORY CONSTRAINTS . . . 89
5.1 Abstract . . . 91
5.2 Introduction . . . 91
5.2.1 Location arc routing problems . . . 91

5.2.2 Periodic capacitated arc routing problem 92
5.3 Mathematical model 93
5.3.1 Problem definition 93
5.3.2 Mathematical model 94
5.4 Location and routing algorithms 100
5.4.1 Location algorithms 100
5.4.2 Allocation 102
5.4.3 Routing 103
5.4.4 Adaptive large-neighborhood search 104
5.5 Test results 105
5.5.1 Parameters 105
5.5.2 Alternative location of one depot 107
5.5.3 Comparison of one and several depots 108
5.5.4 Comparison of the depot-location methods 109
5.6 Conclusion 111

CHAPITRE 6 DISCUSSION GÉNÉRALE 112

CHAPITRE 7 CONCLUSION ET RECOMMANDATIONS 114
7.1 Synthèse des travaux 114
7.2 Limitations de la solution proposée 115
7.3 Améliorations futures 115

LISTE DES RÉFÉRENCES 117

LISTE DES TABLEAUX

Tableau 2.1 Exemple de combinaisons pour une fréquence de 2 et 3 jours par semaine. 22
Tableau 2.2 Exemple de définition de sous-périodes. 25
Tableau 3.1 Values of μ and time limit for *gdbj* instances. 50
Tableau 3.2 Values of μ and time limit for *jpr* instances. 51
Tableau 3.3 Computational results for *gdbj* instances. 52
Tableau 3.4 Computational results for *jpr* instances. 53
Tableau 3.5 Computational results for *gdbj* instances after distance transformation. 53
Tableau 3.6 Maximum number of periods for which models A and B find an optimal solution within the time limit for *gdbj* instances. 54
Tableau 3.7 Maximum number of periods for which models A and B find an optimal solution within the time limit for *jpr* instances. 55
Tableau 3.8 Comparison of routes for *gdbj20* network from models A and B. 55
Tableau 3.9 Comparison of routes for *gdbj19* network from models A and B. 56
Tableau 3.10 Comparison of routes for *jpr8* network from models A and B. 56
Tableau 3.11 Comparison of different vehicle capacities for *jpr6* instance using model B. 57
Tableau 3.12 Percentage reduction in computational time when stopping criterion set to 2% relative gap for *gdbj* instances. 58
Tableau 3.13 Percentage reduction in computational time when stopping criterion set to 2% relative gap for *jpr* instances. 59
Tableau 4.1 Best solution from running the eight operators separately for *gdbj* instances. 82
Tableau 4.2 Best solution from running the eight operators separately for *mine* instances. 82
Tableau 4.3 Percentage of improvement on the initial solution and computation time for the combinations of operators used for *gdbj* instances. 84
Tableau 4.4 Percentage of improvement on the initial solution and computation time for the combinations of operators used for *mine* instances. 85
Tableau 4.5 Percentage of improvement of the combinations against the best solution for *gdbj* instances. 86
Tableau 4.6 Percentage of improvement of the combinations against the best solution for *gdbj* instances. 86

Tableau 4.7 Comparison between three different vehicle capacities : Restricted, regular and unlimited. 87
Tableau 5.1 General information on the *mine* instances tested. 106
Tableau 5.2 Value of the objective function with one depot located at node 0 and at node n. 107
Tableau 5.3 Comparison of the objective function values for one and N depots. . . . 108
Tableau 5.4 Comparison of the objective function values for one and N depots after ALNS. 109
Tableau 5.5 Comparison of the objective function values for the three depot-location methods without ALNS improvement. 110
Tableau 5.6 Comparison of the three depot-location methods after the ALNS improvement. 111

LISTE DES FIGURES

Figure 1.1 Représentation d'un réseau. 2
Figure 1.2 Niveau d'humidité dans une arête de la mine à ciel ouvert. 3
Figure 1.3 Niveau d'humidité et livraison d'eau sans pénurie. 4
Figure 1.4 Niveau d'humidité sur les arêtes A, B et E, a) si on choisit le trajet D-1-2-4-5-D ; b) si on choisit le trajet D-5-4-2-1-D. 5
Figure 2.1 Classement des problèmes du domaine des CARP. 10
Figure 2.2 Deux trajets pour le CARP. 20
Figure 2.3 Différence entre utiliser une solution pour un jour et une solution pour une semaine complète. 20
Figure 2.4 Modèle M1 pour le PCARP. 23
Figure 2.5 Modèle M2 26
Figure 2.6 a. Augmentation de la quantité de matériel dans les applications du PCARP. b. Réduction de la quantité de matériel dans le problème PCARP avec contraintes de gestion de stocks. 33
Figure 3.1 *a)* Representation of the road network in an open-pit mine. *b)* Directed network with the artificial arc $(0, 0)$ at the depot. 40
Figure 3.2 Humidity inventory of edge (i, j). 41
Figure 3.3 Discretization of time. 41
Figure 3.4 Example of different truck speeds 47
Figure 3.5 Representation of network *gdbj19*. 49
Figure 4.1 a) Humidity level of edge $[i, j]$ in a continuous time horizon. b) Humidity level of edge $[i, j]$ after the time horizon is discretized in equal periods. 67
Figure 4.2 a. Satellite view of one of the mines used to determine instances *mine*. b. The network that correponds to the pit shown in a. 80
Figure 4.3 Percentage of improvement of the cost of the initial solution when using the eight operators separately for *gdbj* instances. 81
Figure 4.4 Percentage of improvement of the cost of the initial solution when using the eight operators separately for *mine* instances. 83
Figure 5.1 Humidity level of edge $[i, j]$ 94
Figure 5.2 Evaporation factor e^{tc} for the three scenarios. 110

LISTE DES SIGLES ET ABRÉVIATIONS

ALNS	Adaptive Large Neighborhood Search
ARP	Arc Routing Problem
CARP	Capacitated Arc Routing Problem
CARPIF	Capacitated Arc Routing Problem with Intermediate Facilities
CARP-RP	Capacitated Arc Routing Problem with Refill Points
CARPTW	Capacitated Arc Routing Problem with Time Windows
CPP	Chinese Postman Problem
CVRP	Capacitated Vehicle Routing Problem
DCARP	Directed Capacitated Arc Routing Problem
IRP	Inventory Routing Problem
LARP	Location Arc Routing Problem
LRPP	Location Rural Postman Problem
MCARP	Mixed Capacitated Arc Routing Problem
M-CARP	Multi-depot Capacitated Arc Routing Problem
OCARP	Open Capacitated Arc Routing Problem
PCARP	Periodic Capacitated Arc Routing Problem
PCARP IC	Periodic Capacitated Arc Routing Problem with Inventory Constraints
RPP	Rural Postman Problem
SCARP	Stochastic Capacitated Arc Routing Problem
UCARP	Undirected Capacitated Arc Routing Problem
VND	Variable Neighborhood Descent
VNS	Variable Neighborhood Search
VRP	Vehicle Routing Problem

CHAPITRE 1

INTRODUCTION

Les particules de poussière qui se retrouvent dans l'air posent un grave problème de santé pour les travailleurs des mines et les personnes vivant dans les régions voisines. Les plus petites particules atteignent les voies respiratoires, et sont donc très nocives (Tian *et al.*, 1996). D'autre part, les particules plus grosses réduisent la durée de vie des machines et des véhicules et peuvent aussi causer des accidents en raison de la visibilité réduite. La poussière peut nuire sensiblement à la végétation et affecter l'agriculture locale (Greening, 2011). Les préoccupations environnementales augmentent lorsqu'on retrouve des particules métalliques mélangées à la poussière. D'où l'importance de supprimer la poussière dans les activités minières comme le forage, le dynamitage, le dépôt des matériaux et le transport.

Nous nous concentrons sur la suppression, dans les mines à ciel ouvert, de la poussière due à la circulation des véhicules le long des chemins de terre.

Les techniques de suppression de la poussière comprennent des solutions à long terme comme le pavage des chemins ou l'addition de gravier. Leur coût élevé peut être justifié par l'utilisation à long terme des routes. Cependant, le caractère temporaire des routes dans les mines à ciel ouvert ne permet pas une telle solution.

Parmi les substances utilisées dans la suppression de la poussière, comme le chlorure de calcium ou le chlorure de magnésium, l'eau reste la moins chère et la plus facile à utiliser (FCM, 2005). Le seul inconvénient est qu'elle doit être appliquée périodiquement en raison de sa faible rétention au sol et de l'évaporation continue.

L'eau s'applique en utilisant des camions-citerne qui parcourent les chemins d'une mine. Ils se remplissent à un dépôt central pour commencer un nouveau trajet. L'objectif est de trouver un itinéraire le moins coûteux pour les camions de telle sorte que l'efficacité de rétention des particules de poussière soit maximisée. Parce que l'activité principale a lieu sur les chemins d'un réseau, et comme on utilise un véhicule pour réaliser l'activité principale, ce problème d'application appartient à la branche de la recherche opérationnelle appelée problèmes de tournées de véhicules. D'autre part, la gestion du niveau d'humidité dans les routes joue un rôle important dans la réduction des coûts de pénurie associés à une quantité réduite

d'humidité. On a donc un problème qui exige deux types de décisions : décider quelles sont les routes à desservir et la quantité d'eau à fournir à chacune. La première décision est liée aux problèmes de tournées sur les arcs et implique l'élaboration d'un itinéraire à suivre par le camion-citerne pour atteindre les arêtes choisies. La seconde décision est un problème de gestion des stocks, dans laquelle, chaque arête du réseau est un client avec un niveau d'inventaire qui correspond à la quantité d'humidité et qui doit être réapprovisionné.

Même si on se concentre sur les deux principales décisions, il y a d'autres considérations impliquées dans le problème de l'arrosage des chemins, telles que le nombre de camions-citerne à utiliser, le nombre de dépôts d'eau à installer et le choix des sites où les placer dans le cas où il y en a plus d'un.

1.1 Décisions de routage et gestion de stocks

La combinaison de deux décisions, gestion de stocks et routage, a été étudiée pour des problèmes de tournées sur les nœuds. Cependant, il n'existe pas d'applications dans la littérature concernant les problèmes de tournées sur les arcs. Pour illustrer comment la gestion de stocks et le routage sont combinés pour obtenir une solution au problème d'arrosage des routes, considérer la figure 1.1. Le nœud «D» représente le dépôt et les arêtes A à H représentent les segments de routes d'une mine à ciel ouvert.

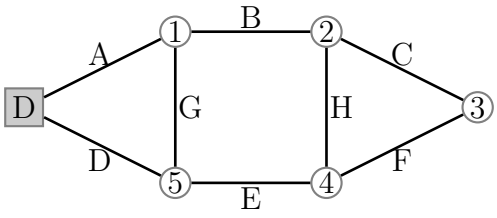

Figure 1.1 Représentation d'un réseau.

Chaque arête a un niveau d'humidité qui peut être représenté avec un modèle de gestion de stocks tel qu'on montre dans la figure 1.2. Q représente la quantité d'eau à livrer et h_m représente le niveau d'humidité requis pour assurer la rétention de la poussière. Si le niveau d'humidité se trouve sous la ligne h_m, il existe une pénurie. Le niveau d'humidité ne peut pas être négatif.

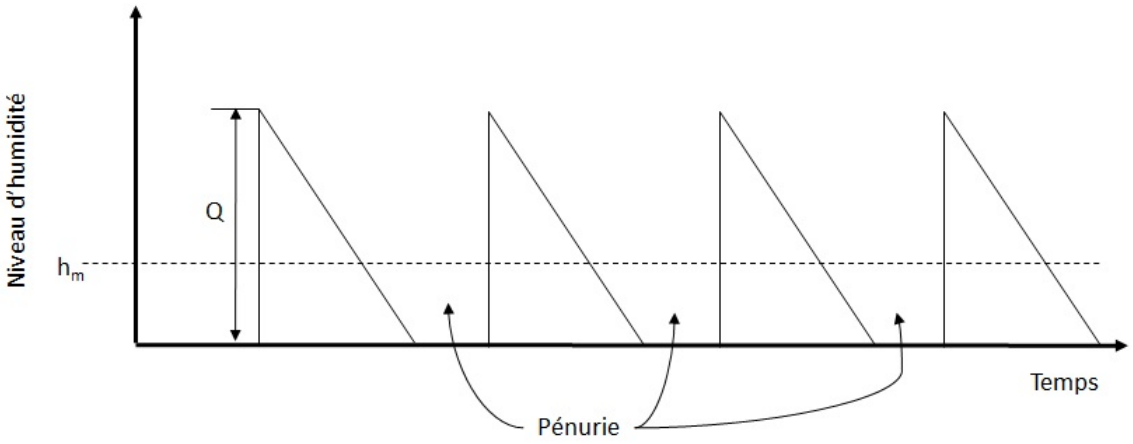

Figure 1.2 Niveau d'humidité dans une arête de la mine à ciel ouvert.

On pose les hypothèses suivantes qui correspondent à un modèle de réapprovisionnement par quantité économique de Silver *et al.* (1998) :

- La demande peut être estimée à l'aide des lectures d'évaporation d'eau dans un bac. Selon Neulicht et Shular (1998), l'évaporation dans un bac est la meilleure façon d'estimer l'évaporation de l'eau au sol, mais les lectures sont effectuées une fois par jour. Molina-Martínez *et al.* (2006) ont développé un moyen de simuler l'évaporation horaire du bac. Ces taux d'évaporation s'appliquent à toutes les routes de la mine et dépendent du moment de la journée. La consommation d'eau est également affectée par le volume du trafic. À chaque arête est attribuée un facteur d'évaporation différent selon le nombre de camions circulant sur la route correspondante. Les deux facteurs, l'évaporation et le volume du trafic, sont utilisés pour estimer le taux de consommation de l'humidité.
- La quantité d'eau à livrer ne doit pas forcément être un nombre entier et peut être différente à chaque visite. Il existe une quantité minimum d'humidité sur les arêtes quand le niveau est 0. Il existe un niveau d'humidité maximum qui correspond à la quantité d'eau qui rend les routes boueuses et glissantes.
- Il n'y a pas un coût de possession de matériel pour la quantité d'humidité dans les routes. Les coûts à considérer sont les coûts de livraison qui correspond aux coûts de routage et les coûts de pénurie. Les coûts ne changent pas par rapport au temps.
- Il existe un seul matériel livré, soit l'eau.
- On considère que le réapprovisionnement est fait sans délai au début du remplissage.
- On considère un horizon de temps qui correspond à un quart de travail divisé en périodes de temps de durée identique. Une période de temps est la quantité de temps qu'un camion-citerne prend afin de couvrir une distance constante à une vitesse constante.

Par exemple, un camion roulant à 20 km/h peut couvrir une distance constante de 300 m en environ 1 minute (54 secondes).

– La pénurie est considérée et pénalisée.

L'objectif de ce modèle de gestion de stocks est de réduire le coût de pénurie et de trouver la façon de livrer l'eau de telle sorte que le niveau d'humidité soit au-dessus du niveau h_m. On montre cette situation à la figure 1.3. On montre, dans cet exemple, que la quantité d'eau peut varier et peut être livrée à différents intervals de temps, mais le nievau d'humidité est toujours inférieur à la quantité maximale d'humidité, h_{max}.

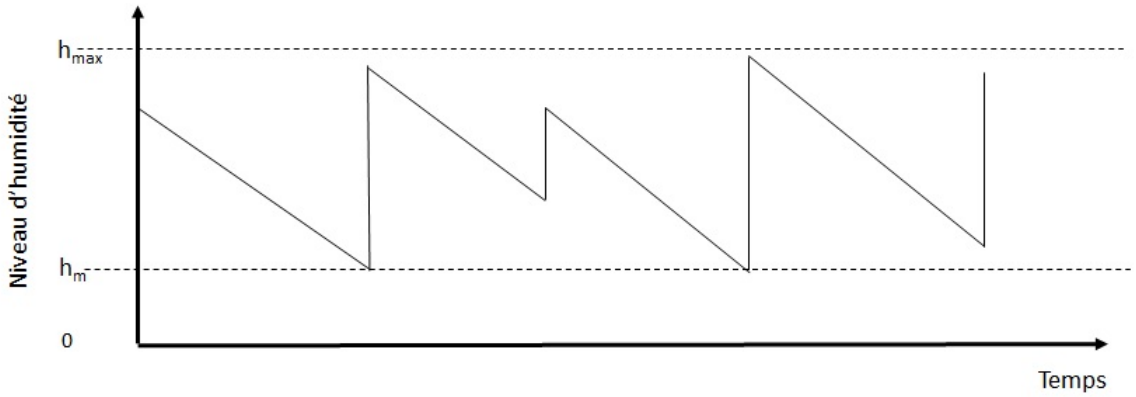

Figure 1.3 Niveau d'humidité et livraison d'eau sans pénurie.

En considérant les arêtes de la figure 1.1, on peut montrer l'implication de choisir un trajet parmi les arêtes qui minimisent le coût de pénurie à cause du niveau d'humidité sur les routes. Supposons qu'on désire arroser les arêtes A, B et E. Le niveau d'humidité est montré à la figure 1.4. On peut parcourir les arêtes dans la direction D-1-2-4-5-D ou dans la direction D-5-4-2-1-D. La figure 1.4a montre le premier trajet où l'arête A est arrosée au temps t_1, l'arête B est arrosée au temps t_2 et après traversé l'arête H sans l'arroser, on arrose E au temps t_3. Au temps t_3, l'humidité a été déjà évaporée sous le niveau h_m ce qui résulte en une pénurie pour E. À la figure 1.4b, on considère le deuxième trajet qui commence par traverser l'arête D sans l'arroser. On arrose E au temps t_1, on traverse H et finalement on arrose B au temps t_2 et A au temps t_3. Au moment d'arroser B et A, on a déjà une pénurie. On va choisir le trajet en fonction de l'aire totale sous h_m pour les trois arêtes et de la priorité de chacune de ce trois arêtes. Dans les deux trajets, la quantité livrée à chacune des arêtes est différente.

La figure 1.4 montre comment les décisions de routage peuvent affecter le coût d'inven-

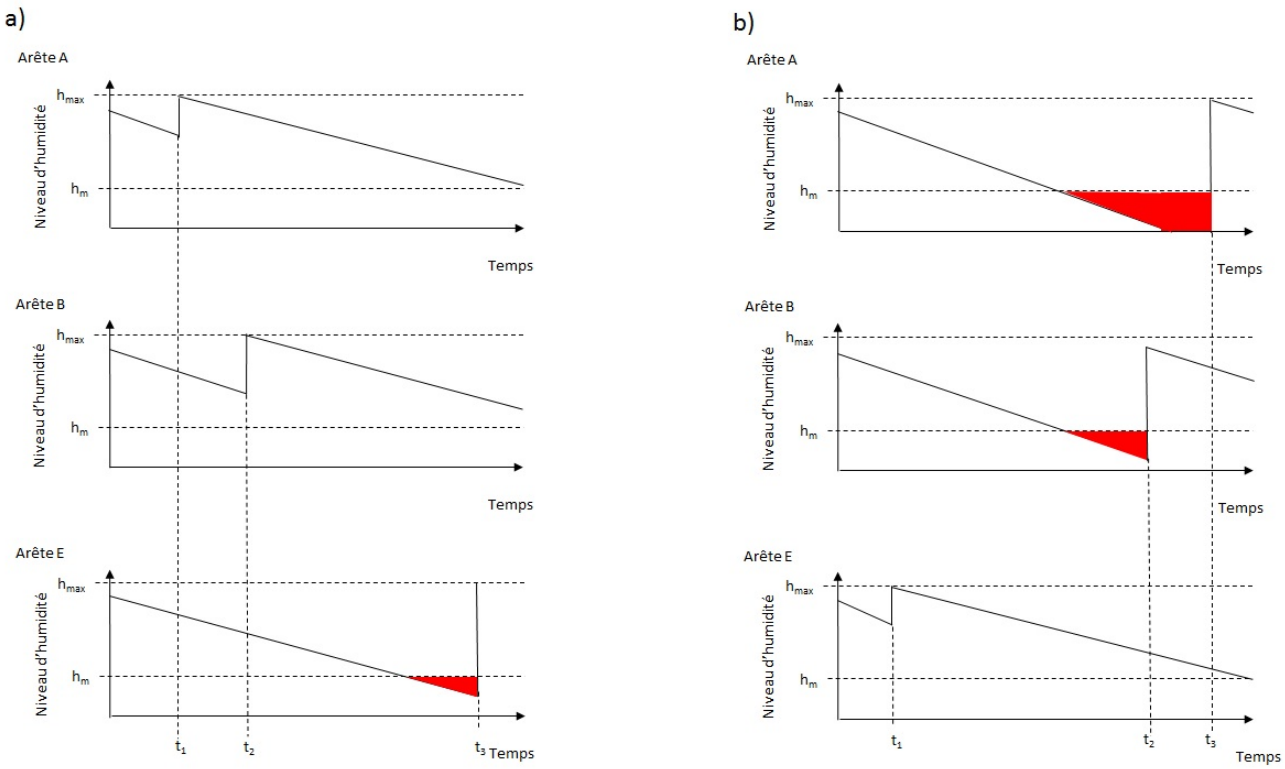

Figure 1.4 Niveau d'humidité sur les arêtes A, B et E, a) si on choisit le trajet D-1-2-4-5-D ; b) si on choisit le trajet D-5-4-2-1-D.

taire. D'autres décisions qui peuvent être prises sont le nombre de camions nécessaires pour arroser les arêtes du réseau, la quantité d'eau à arroser sur chacune d'elles et la localisation des dépôts d'eau, dans le cas où y en a plus d'un. Tous ces éléments peuvent affecter le trajet des camions et le niveau d'humidité des arêtes.

Le réapprovisionnement de l'inventaire sur chaque arête montre la nature périodique de ce problème. Quand on finit un trajet, il faut remplir le camion et décider quelles arêtes arroser sur le trajet suivant. À la figure 1.4a, par exemple, l'arête E vient d'être arrosée, il ne sera pas nécessaire de l'arroser de nouveau après le remplissage du camion au dépôt. Par contre, les arêtes A et B peuvent être arrosées ainsi qu'une autre arête du réseau prenant en compte la capacité du camion. L'arrosage continuera jusqu'à la fin de l'horizon de temps.

Ce problème est une application des problèmes périodiques de tournées sur les arcs, avec la caractéristique que chaque arête a besoin d'une quantité de matériel plus d'une fois durant l'horizon de temps, ce qui ressemble à un problème de gestion de stocks. On fait l'hypothèse

qu'il est possible de programmer une tournée sur le réseau, qui débute et finit au dépôt, pour servir les arêtes de telle façon qu'on minimise les coûts de routage et de la pénurie du niveau d'humidité. Le nombre de combinaisons possibles augmente en raison de la taille du réseau. Dans les chapitres suivants, on décrit les approches pour trouver une ou plusieurs tournées possibles pour ce problème.

Le chapitre 2 présente une revue de la littérature qui porte sur les problèmes de tournées sur les arcs avec contraintes de capacité. On porte une attention particulière aux problèmes périodiques et à leurs applications. Le chapitre 3 décrit le modèle mathématique créé pour le problème de suppression de la poussière dans les mines à ciel ouvert. Le chapitre 4 présente un algorithme heuristique qui est capable de résoudre le problème d'arrosage de chemins de terre pour des réseaux de grande taille. Le chapitre 5 examine la façon de distribuer des réservoirs d'eau le long de la mine pour favoriser l'arrosage des chemins. Finalement, nos conclusions sont présentées au chapitre 6.

CHAPITRE 2

REVUE DE LITTÉRATURE

Dans le domaine des problèmes de tournées, ceux dont l'activité principale se produit sur les arcs du réseau sont connus comme *Arc Routing problems* (ARP). Un résumé de l'histoire de la recherche sur les ARP se trouve dans Assad et Golden (1995). Une revue des ARP est divisé en deux parties : Eiselt *et al.* (1995b) présentent une étude sur le problème de postier chinois (CPP), un ARP qui vise à trouver la tournée de longueur minimale qui visite chaque arête d'un réseau au moins une fois, et Eiselt *et al.* (1995a) présentent une revue sur le problème du postier rural (RPP), un ARP similaire au CPP où seulement un sous-ensemble d'arêtes ont besoin de service. Une bibliographie annotée a été presentée dans Laporte et Osman (1995). Les premièrs problèmes d'ARP avaient pour objectif de trouver la façon de parcourir les arêtes d'un réseau. Des applications réelles telles que la collecte de déchets ou l'épandage sur les rues en hiver ont mené à l'incorporation de contraintes de capacité sur les véhicules. Ces dernières exemples font partie des problèmes de tournées sur les arcs avec contraintes de capacité (CARP).

2.1 Problèmes de tournées sur les arcs avec contraintes de capacité

On appelle CARP l'ensemble des problèmes de tournées sur les arcs avec contraintes de capacité. Ils ont été introduit par Golden et Wong (1981). Les arcs d'un réseau ont une demande spécifique tandis que les véhicules qui donnent le service ont une capacité limité. Lorsque la somme des ressources accumulées lors de la visite des arcs d'une tournée d'un véhicule atteint la capacité maximale du véhicule, ce dernier doit retourner au dépôt. Les caractéristiques du CARP sont :

- Les arcs ont une demande $q_{ij} \geq 0$. Chaque arc avec $q_{ij} > 0$ doit être visité par un véhicule. Les arcs sans demande peuvent être traversés pour se déplacer vers des arcs qui ont besoin de services.
- La somme de la demande des arcs dans une tournée ne doit pas dépasser la capacité Q_v du véhicule, i.e., $\sum_{ij} q_{ij} \leq Q_v$.
- L'objectif est de minimiser le coût total des trajets d'un ou plusieurs dépôts vers les arcs avec retour aux dépôts à la fin.

Dans cette section, une revue générale de littérature sur le CARP sera présentée. Certains modèles et méthodes proposés par des auteurs du CARP sont utilisés également par des auteurs du PCARP, d'où l'importance de les présenter et d'en donner une brève explication. Deux revues de littérature sur le sujet du CARP ont été publiées récement, l'une par Wohlk (2008) et l'autre par Corberán et Prins (2010).

2.1.1 Classement de problèmes de CARP

La figure 2.1 montre un classement des problèmes CARP existants. Ce classement et les définitions sont tirés de la revue des problèmes CARP effectuée par Corberán et Prins (2010).

Classement par type de graphe

Le CARP non orienté, ou UCARP, est basé sur un graphe $G = (N, E)$ où N est l'ensemble de nœuds et E est l'ensemble d'arêtes, qui peuvent se traverser dans les deux sens. Par contre, le CARP orienté, ou DCARP, est basé sur un graphe orienté $G = (N, A)$, où A est l'ensemble des arcs qui peuvent être traversés dans un seul sens. Le CARP mixte, ou MCARP, est une approche plus réaliste, car il combine des arcs et des arêtes dans un graphe $G = (N, E \cup A)$. Le MCARP peut être utilisé pour modéliser une situation où certains segments de rue peuvent se traverser dans un seul sens et d'autres dans les deux sens.

Pour le UCARP, DCARP et MCARP, on a un ensemble d'arcs ou arêtes qui ont besoin de service identifié par $A_r \subseteq A$ ou $E_r \subseteq E$. L'objectif est de trouver la façon la moins chère de parcourir le réseau pour servir les éléments de E_r ou A_r.

Classement par type d'application

Le CARP stochastique ou SCARP (Fleury *et al.*, 2004) ; (Fleury *et al.*, 2005), a été développé pour des problèmes où la demande sur chaque segment de route est aléatoire et on utilise des variables aléatoires pour calculer la valeur de la fonction objectif. Les auteurs ont développé un algorithme genetique qui sera expliqué dans la section suivente. Des exemples plus récents incluent le CARP avec incertitude (Mei *et al.*, 2010), où la demande et le coût de parcours sont des variables aléatoires, et le CARP avec demandes stochastiques (Laporte *et al.*, 2010), qui est une application à la collecte des déchets avec des quantités aléatoires. Les auteurs ont développé un algorithme *Adaptive Large Neighborhood Search* (ALNS) qui permet de modifier la solution par un processus itératif dans lequel on utilise un ensemble d'opérateurs d'élimination et d'insertion.

Dans le problème multi-dépôt ou M-CARP (Amberg *et al.*, 2000), le réseau a M dépôts desquels un ensemble de véhicules partent, servent les arêtes requises une seule fois et finissent le trajet au même dépôt. Une variante du problème M-CARP est le CARP avec dépôts intermédiaires, ou CARPIF, (Ghiani *et al.*, 2001) dans lequel les véhicules visitent plusieurs stations pour charger ou décharger leur contenu. Une autre variante est le CARP avec points de recharge, CARP-RP, (Amaya *et al.*, 2007). Dans cette application, des véhicules de service tracent les lignes sur les rues et sont rechargés à certains nœuds du réseau par des véhicules de recharge. Ces derniers retournent vers le dépôt. Donc il faut programmer des rendez-vous entre les deux types de véhicules. Une application similaire est le CARP avec dépôts mobiles (Del Pia et Filippi, 2006). C'est une application de collecte des déchets où on a deux types de camions : de gros camions qui ne peuvent traverser toutes les rues et de petits camions qui ont une faible capacité. Pour éviter le retour des petits camions vers le dépôt, il faut organiser des rendez-vous avec les grands camions qui peuvent servir de dépôts.

Le CARP ouvert, OCARP, a été introduit par Usberti *et al.* (2011) pour des applications où il n'existe pas de dépôt dans le réseau et il n'est pas nécessaire de créer des trajets qui forment un cycle. Les véhicules peuvent débuter et finir à différents nœuds du réseau. Deux applications sont mentionnées et modélisées comme un CARP : le problème de lecture de compteurs, qui a été traité dans (Moreira *et al.*, 2007) comme un problème de postier chinois et le problème de déterminer un chemin de coupe de différents pièces à partir d'une grande plaque métallique en utilisant plus d'un outil (Stern et Dror, 1979). Ce dernier problème est traité comme un problème de postier rural.

Le problème de CARP avec fenêtres de temps, ou CARPTW, a des contraintes pour débuter et finir le service sur chacun des arcs (Mullaseril, 1997). Un problème similaire a été traité par Tagmouti *et al.* (2007), où il existe une pénalité dans le coût total si le service débute plus tôt ou plus tard que le temps souhaité.

Le CARP avec demande au *dead-heading* a été introduit par Kirlik et Sipahioglu (2012) pour modéliser des situations où les arêtes qui n'ont pas besoin de service (dead-heading) ont une demande qui doit être considéré pour la contrainte de capacité du véhicule. On retrouve des applications dans la détection des mines terrestres, les activités de recherche et sauvetage, et la patrouille faite par des robots qui utilisent une quantité d'énergie en traversant les arcs qui ne sont pas desservis. On considère l'énergie du robot comme la contrainte de capacité.

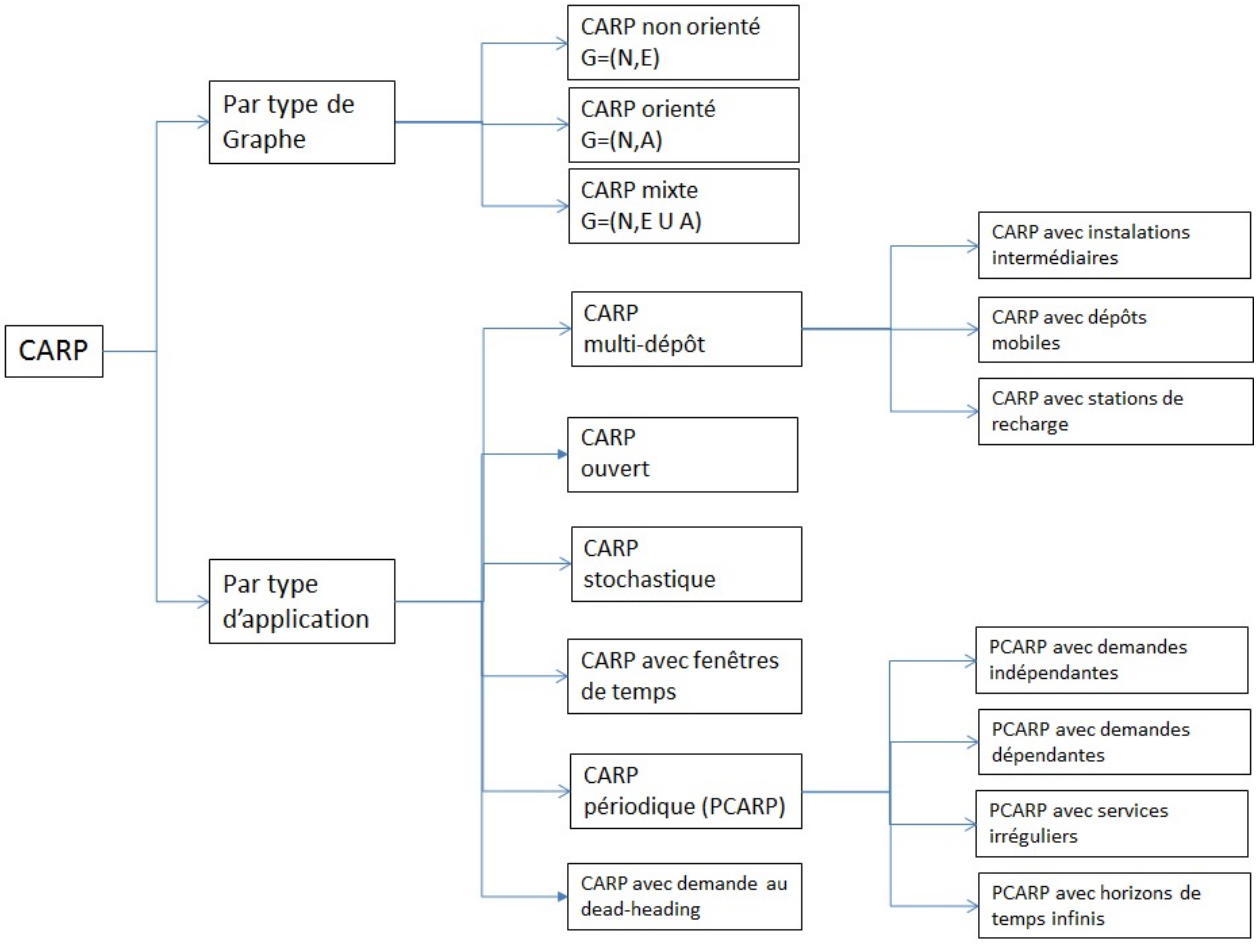

Figure 2.1 Classement des problèmes du domaine des CARP.

Le CARP périodique ou Periodic Capacitated Arc Routing Problem (PCARP) par ses initiales en anglais, est le problème où les segments de route A_r ou E_r nécessitent une fréquence de service et il faut les visiter une fois ou plus pendant un horizon de temps.

Classification des applications du PCARP

Cette classification a été faite par Lacomme *et al.* (2005). Les applications du PCARP peuvent être divisées en deux types :

Type A. Ce sont les problèmes dont la demande est indépendante de la période. C'est-à-dire que chaque fois qu'un arc est visité, la demande, la durée et le coût de service sont les mêmes. Ils ne changent pas, même si le temps entre chaque visite est différent. Un exemple d'application est l'inspection de lignes électriques.

Type B. Ce sont les problèmes dont chaque arc génère une certaine quantité de demande pour chaque période. Ainsi, la durée et le coût de service peuvent être différents chaque fois qu'on visite un arc.

Les problèmes de type B peuvent être divisés en : non cycliques (B1) et cycliques (B2). Dans le premier cas, l'horizon de planification peut commencer indépendamment de ce qui s'est passé à la fin du dernier horizon, par exemple, couper la végétation le long des rues. Dans les problèmes cycliques, l'horizon de planification doit prendre en compte la planification du dernier horizon, comme dans la collecte de déchets.

Récemment, Monroy *et al.* (2011) ont ajouté une deuxième division pour les problèmes de type A et B : ceux avec une fréquence constante qui ont un besoin de services réguliers et ceux avec une fréquence différente pour lesquels il est nécessaire de calculer la fréquence des services.

2.1.2 Méthodes de résolution du CARP

Le premier modèle mathématique pour le CARP a été présenté par Golden et Wong (1981). L'objectif est de minimiser la distance totale parcourue. Les auteurs montrent que le problème est NP-dur et proposent un algorithme heuristique qui sera expliqué à la section suivante.

Comme il sera montré dans les chapitres suivants, cette thèse se concentre principalemnet sur l'utilisation de méthodes heuristiques et métaheuristiques. D'où l'importance d'exposer ces méthodes pour le CARP. Néanmoins, il est important de souligner le travail qui a été réalisé en termes d'algorithmes exacts. Un algorithme de *Branch and Bound* a été proposé dans Hirabayashi *et al.* (1992). Cet algorithme a été testé pour 10 exemplaires de problèmes avec 15 à 50 arcs qui nécessitent du service. Un algorithme de plans de coupe (*cutting plane*) a été proposé par Belenguer et Benavent (2003) pour trouver la borne inférieure de quatre ensembles connus d'exemplaires CARP. Amaya *et al.* (2007) ont formulé un modèle de programmation linéaire en nombres entiers pour le CARP-RP. Ils ont utilisé une méthode de plans de coupe pour résoudre ce nouveau problème. Différentes méthodes pour trouver des bornes inférieures pour le CARP incluent une méthode de coupe et de génération de colonnes combinées avec un problène de partition d'ensembles par pour le CARP (Bartolini *et al.*, 2013b) ; un algorithme cut first, branch-and-price second (Bode et Irnich, 2012). Martinelli *et al.* (2013) ont développé des algorithmes pour trouver des bornes inférieures et supérieures pour des exemplaires de grande taille du CARP (200 nœuds et 300 arêtes). Un algorithme

exact pour le CARP avec demande au dead-heading a été proposé par Bartolini *et al.* (2013a).

Afin d'utiliser les méthodes développées pour les problèmes de tournés sur les nœuds avec contraintes de capacité (CVRP), Baldacci et Maniezzo (2006) et Longo *et al.* (2006) ont proposé la transformation du problème CARP en un problème CVRP équivalent en changeant le réseau $G = (N, E)$ par un réseau équivalent $G' = (N', E')$. Ces deux méthodes sont basées sur la transformation proposée par Pearn *et al.* (1987) dans laquelle le graphe résultant a $3R + 1$ sommets, où R est le nombre d'arêtes requises dans le graphe original. La transformation proposée par Baldacci et Maniezzo consiste à répliquer en G' les sommets de G autant de fois que le nombre d'extrémités des arêtes E requises et ajouter un sommet pour le dépôt. Pour avoir $2R + 1$ sommets. Ensuite, les arêtes E' sont ajoutées pour rendre le graphe G' connexe. Le coût des arêtes E' est calculé en utilisant les plus courts chemins sur le graphe G et une solution réalisable du problème comme borne supérieure. Le problème est ensuite résolu en utilisant un algorithme de *Branch and Cut*. Cette méthode a été utilisée pour résoudre des problèmes ayant jusqu'à 98 arêtes requises. La transformation proposée par Longo *et al.* (2006) résulte en un graphe G' avec $2R+1$ sommets et les arêtes R' requises qui correspondent aux arêtes R du graphe G et qui doivent être traversées dans la solution. Ensuite les auteurs, résolvent le problème en utilisant un algorithme de type *Branch and cut and price*. Ils ont obtenu la solution optimale pour des exemplaires déjà connus du CARP.

Méthodes heuristiques et métaheuristiques pour le CARP

Était donné la complexité du CARP, les auteurs qui ont traité ce problème ont développé des méthodes heuristiques pour résoudre des exemplaires de grande taille. On présentera les principaux algorithmes.

Algorithme **Construct and Strike** (Christofides, 1973).
Cet algorithme a été formulé pour le problème du postier chinois avec contraintes de capacité. Dans un graphe :

1. Construire un cycle qui sert un sous-ensemble d'arêtes en tenant compte de deux propriétés :
   (a) La contrainte de capacité doit être respectée.
   (b) Le graphe doit demeurer connexe après qu'on ait enlevé les arêtes servies.

2. Résoudre un problème du couplage entre les nœuds de degreé impair de telle façon que tous les nœuds du graphe aient un degré pair.
3. Répéter l'algorithme jusqu'à ce que tous les arêtes se trouvent dans un cycle.

Après l'introduction du CARP (Golden et Wong, 1981), plusieurs algorithmes heuristiques ont été utilisés pour traiter des applications spécifiques. La plupart des algorithmes récemment développés sont basés sur les premières heuristiques proposées. On va les décrire pour pouvoir y référer dans la description des algorithmes plus complexes. Une brève description de ces algorithmes se trouve dans Wohlk (2005).

Algorithme **Augment Merge** (Golden et Wong, 1981).

1. Construire un cycle qui relie le dépôt à chaque arête qui a besoin de service, appelée *tâche*, et trier en ordre décroissant des coûts de routage.
2. Déterminer, parmi les cycles trouvés, et en commençant par le cycle le plus grand, s'il est possible de servir un petit cycle dans un grand cycle. On appelle cette phase *augmentation*.
3. Si la capacité du véhicule est suffisante, fusionner deux cycles qui occasionnent la plus grande économie. Cette phase est répétée jusqu'à ce qu'on atteigne la capacité du véhicule. On appelle cette phase *unification*.

Algorithme **Path Scanning** (Golden *et al.*, 1983)
On débute avec une route vide et on ajoute un arc (i, j) à la route en utilisant un des cinq critères suivants :

1. Minimiser la distance par unité de demande (coût/demande).
2. Maximiser la distance par unité de demande (coût/demande).
3. Minimiser la distance du nœud j au dépôt (cycles plus courts).
4. Maximiser la distance du nœud j au dépôt (cycles plus longs).
5. Si la capacité du véhicule est inférieure à la moitié, maximiser la distance du nœud j au dépôt, sinon, minimiser la distance.

Si la demande de l'arc suivant dépasse la capacité du véhicule, on trouve le plus court chemin au dépôt et on commence la route suivante. Finalement, on construit une solution avec chacun des critères 1 à 5 et on choisit celle de coût minimum comme solution finale.

Algorithme **Route first-cluster second** pour le CARP (Ulusoy, 1985)
Cet algorithme est appliqué sur un graphe $G = (N, A)$ où $A_r \subseteq A$, est l'ensemble des arcs qui ont besoin de service. G se transforme comme suit :

1. Trouver un graphe G^E qui contienne un tour d'Euler géant sans prendre en compte les contrainte de capacité. S'il y a des arcs répétés de A_r, ils se transforment en arcs sans service.
2. À partir de G^E, trouver un graphe G^* comme suit :
   (a) Le premier nœud est le premier nœud de G^E et les nœuds suivants sont les arêtes de G^E.
   (b) Si le dernier arc de G^* est répété dans G^E, il est éliminé.
   (c) Les arcs de G^* sont des trajets réalisables dans le graphe original G.
3. Calculer les coûts c_{ij} des arcs de G^* en ajoutant les coûts des arcs correspondants dans G.
4. Calculer le plus court chemin du nœud 1 au dernier nœud dans G^*.
5. Déterminer les trajets de chaque arc dans le plus court chemin. Éliminer les trajets qui n'ont pas d'arcs de A_r.

Une adaptation de cet algorithme a été utilisé dans Kirlik et Sipahioglu (2012) pour le problème du CARP avec demande au dead-heading. La principale différence est que les arcs non servis doivent être considérés pour calculer les arcs réalisables dans G^*, et ce processus est fait après le calcul de chaque trajet.

Algorithme **insertion parallèle** (Chapleau *et al.*, 1984)

1. Déterminer le nombre de routes nécessaires.
2. Pour chaque route, trouver un arc pour débuter.
3. Construire les routes en parallèle en utilisant les processus suivants de manière alternative :
   – Pour un arc donné, trouver la route où il peut être insérer de manière à minimiser le coût d'insertion.
   – Pour une route donnée, déterminer l'arc à insérer.
4. Arrêter quand tous les arcs ont été utilisés.

Algorithme **Cluster first-route second** (Benavent *et al.*, 1990).
Dans un graphe $G = (N, A)$ à être parcouru par k véhicules :

1. Déterminer k nœuds, appelés *centres*, les plus éloignés les uns des autres et les plus éloignés du dépôt.
2. Pour chaque centre, sélectionner les arêtes les plus proches de façon séquentielle sans dépasser la capacité du véhicule. Ces ensembles d'arêtes sont appelés *clusters*.
3. Si toutes les arêtes requises ont été sélectionnées, avancer à l'étape de formation de trajets, sinon faire des échanges d'arêtes d'un cluster à l'autre jusqu'à avoir toutes les arêtes affectées à un cluster.
4. Pour chaque cluster, créer un trajet qui débute et finit au dépôt.

2.1.3 Métaheuristiques pour le CARP

Dans cette section on décrit les métaheuristiques utilisées pour le CARP. On n'inclure pas les méthodes pour le CARP périodique qui seront présentées de façon détaillée dans la section suivante.

Recherche tabou

La recherche Tabou utilise des techniques de recherche locale pour explorer un voisinage de solutions. Pour éviter de rester dans une région réduite de l'espace de solution, certains mouvements sont interdits et conservés dans une liste tabou. Seulement une petite partie des mouvements sont inclus dans la liste tabou et sont conservés là pendant un temps relativement courts (Glover, 1986).

Hertz *et al.* (2000) ont développé un méthode de recherche tabou pour le CARP appelée *CARPET*. L'objectif est de minimiser la distance parcourue mais des solutions non réalisables sont admises avec une pénalité. Le processus d'amélioration (POSTOPT et SHORTEN) des solutions permet de réduire le nombre d'arcs sans demande et de grouper les arcs de service dans chaque cycle. Le voisinage est construit par l'addition ou la suppression d'un arc qui a besoin de service (processus ADD et DROP). Tous ces processus ont été présentés par Hertz *et al.* (1999).

Un algorithme de type Route First – Cluster Second a été utilisé par Amberg *et al.* (2000) pour le CARP avec dépôts multiples. Ils créent une route géante sans prendre en compte la contrainte de capacité et pour la phase de Cluster, ils utilisent une méthode d'arbre de poids minimal avec contraintes de capacité pour chaque véhicule disponible dans chaque dépôt. La

solution est améliorée en utilisant des heuristiques de recuit simulé et de recherche tabou. La liste tabou, au lieu de contenir solutions déjà utilisées, garde les mouvements qui ont formé de telles solutions.

Brandao et Eglese (2008) ont proposé un algorithme de recherche tabou complètement déterministe pour le CARP. Ils utilisent deux méthodes d'insertion et une méthode d'échange pour la recherche locale. La longueur de la liste tabou passe de $E/2$ pour $E/6$, dans une phase subséquente, où E est le nombre d'arêtes du réseau. Ils ont testé cet algorithme sur des exemplaires d'environ 250 sommets et 375 arêtes.

Recherche à voisinage variable (VNS)

Cette technique vise à effectuer une recherche dans un voisinage spécifique pour trouver le minimum local et utilise une stratégie pour effectuer la recherche sur un voisinage différent. Une variante est la descente à voisinage variable (VND) qui effectue une recherche locale dans un voisinage jusqu'à trouver un minimum local, puis le voisinage est changé et le processus est répété. La VND applique une méthode de descente dans différents voisinages afin de trouver l'optimum local pour chacun d'eux (Hertz et Mittaz, 2001).

Une méthode de descente à voisinage variable (VND) a été proposée par Hertz et Mittaz (2001). La VND appliquée au CARP vise à minimiser la distance totale. L'espace S de solutions contient toutes les combinaisons qui servent les arrêtes requises et respectent la contrainte de capacité. Le voisinage $N(s)$ d'une solution $s \in S$ a été trouvé en applicant les processus *SWITCH* (modification de l'ordre pour servir les arcs requis), *CUT* (division d'un trajet non réalisable en trajets réalisables), et *SHORTEN* (réduire les trajets sans arcs requis) dans cet ordre à un ensemble de trajets fusionnés. Les résultats ont été comparés avec deux méthodes de recherche tabou et les algorithmes heuristiques décrits à la section précédente.

Un algorithme de recherche locale guidée a été proposé par Beullens *et al.* (2003). L'algorithme utilise une recherche locale composée de six mouvements dans la même route et dans des routes différentes. Une procédure de recherche locale guidée sélectionne les arêtes avec le coût de *dead-heading* le plus élevé pour effectuer la prochaine recherche locale. La procédure est répétée jusqu'à ce qu'un nombre maximum d'itérations soit atteint.

Del Pia et Filippi (2006) ont utilisé un algorithme de descente à voisinage variable pour

le problème du CARP avec dépôts mobiles. L'algorithme a un processus de rendez-vous pour les deux types de camions (gros et petits) de telle façon qu'on respecte la contrainte de capacité pour les gros camions et on minimise le temps pris par les camions pour dévier de leur route originale. Les voisinages ont été trouvés par l'échange d'arcs d'un trajet à l'autre et par le processus *SWITCH - CUT -SHORTEN* de Hertz et Mittaz (2001).

Un algorithme VNS pour le CARPIF a été présenté par Polacek *et al.* (2008). L'algorithme utilise alternativement une heuristique gloutonne et *Route first - cluster second* pour générer la solution initiale. Les différents voisinages sont générées par l'échange de deux segments de route différents choisis au hasard. L'algorithme effectue ensuite une recherche locale jusqu'à ce qu'un critère d'arrêt soit atteint.

Algorithmes évolutifs

Ces algorithmes sont basés sur le concept de sélection naturelle. Il existe un ensemble de solutions qui forme une population initiale. Un certain nombre de solutions (parents) sont choisies de cette population et combinées (croisées) afin de créer de nouvelles solutions (descendants). Ces dernières sont ensuite modifiées (mutation). Le processus est répété avec des solutions nouvellement créées jusqu'à ce qu'un critère d'arrêt soit atteint.

Un algorithme génétique pour résoudre des exemplaires du CARP a été proposé dans Lacomme *et al.* (2001). Une partie de la population initiale est construite de façon aléatoire et l'autre partie est construite en utilisant les méthodes heuristiques de «Path-scanning», «Augment-Merge» et la méthode d'Ulusoy, décrites dans la section précédente. Ensuite, deux parents, P_1 et P_2, sont choisis parmi ceux de la population initiale afin de produire deux enfants, C_1 et C_2. Le processus de croisement est fait en prenant une séquence de tâches du P_1. La séquence est insérée dans C_1 en gardant la même position qu'elle avait dans P_1. Le reste des tâches sont insérées en C_1 à partir de P_2 en gardant le même ordre qu'elles avaient dans P_2. Les tâches qui sont déjà insérées sont évitées. Le processus est similaire pour C_2 en échangeant les rôles de P_1 et P_2. Un des enfants est donc sélectionné aléatoirement pour suivre un processus de mutation par recherche locale, et enfin, remplacer l'un des parents. L'algorithme s'arrête quand on atteint un nombre maximum d'itérations. Pour les exemplaires testés en utilisant cet algorithme génétique, on a obtenu des résultats de qualité comparable à ceux obtenus avec la méthode de recherche tabou, CARPET.

Afin de considérer l'incertitude sur la demande, le problème CARP stochastique (SCARP)

a été décrit dans Fleury *et al.* (2005). La quantité de demande sur chaque arc/arête est une variable aléatoire normale $N(q_{ij}, \sigma_{ij})$ avec une borne supérieure égale à la capacité Q_k des véhicules. Les mêmes auteurs ont proposé un algorithme mémétique pour résoudre le SCARP (Fleury *et al.*, 2004). Cet algorithme travaille de façon similaire à l'algorithme génétique mais avec un processus de recherche locale pour améliorer le croisement. Les auteurs ont aussi utilisé des méthodes mathématiques pour traiter les éléments aléatoires du problème comme la demande sur chaque segment de route, le coût total de chaque route et la possibilité de s'arrêter avant de finir le service sur un segment de route lorsque la demande excède la capacité du véhicule.

Hongtao *et al.* (2013) ont utilisé un algorithme génétique avec un processus de perturbation pour le CARP-MD. L'algorithme est similaire à celui proposé par Lacomme *et al.* (2001), sauf que, quand un enfant sélectionné au hasard remplace un parent, il est soumis à un mécanisme de perturbation par un double échange (échange de deux arêtes avec une autre solution), si l'enfant est un clone de l'une des solutions dans le bassin de parents. Ils mettent aussi en place une recherche locale pour les descendants sélectionnés.

Liu *et al.* (2012) ont développé un algorithme génétique pour le CARP avec plusieurs dépôts et véhicules ayant différentes capacités. Les algorithmes utilisés pour former la population initiale (partition, path scanning) doivent être modifiés pour inclure des dépôts multiples et des véhicules hétérogènes. Après que le croisement et la recherche locale soient terminés, l'algorithme recommence avec les deux meilleures solutions et les solutions nouvellement générées. Cette deuxième phase est faite avec une probabilité plus élevée d'appliquer la recherche locale.

Liu et Ray (2012) ont proposé un algorithme mémétique pour le CARP. L'ensemble des solutions initiales est formé en utilisant l'algorithme *path scanning*. Le croisement est fait en affectant des nombres aléatoires aux tâches de deux parents. Pour les enfants, ces nombres représentent l'ordre d'exécution des tâches. Ensuite, ils utilisent un processus de recherche locale en utilisant des méthodes d'insertion et d'échange.

Liu *et al.* (2013) ont dévélopé un algorithme mémétique avec une nouvelle forme de croisement dans laquelle, on prend la séquence de tâches la plus longue pour le plus petit *deadheading*, qui est similaire dans les deux parents P_1 et P_2 et on l'ajoute aux deux enfants C_1 et C_2 au même endroit. Le reste des tâches est copié dans le même ordre de P_1 à C_2 et de P_2 à C_1. Une processus de recherche locale itérée est utilisé si la solution est très proche de celle du meilleur parent dans la population.

Autres algorithmes

Dans Greistorfer (2003) on trouve un algorithme de recherche tabou combiné avec un algorithme de recherche dispersée. Ce dernier algorithme combine les processus de diversification et intensification de façon systematique et non aléatoire (Martí *et al.*, 2006). Un ensemble de solutions initiales est obtenue à l'aide d'une heuristique de construction. Le voisinage est obtenu par une combinaison des opérations d'insertion et d'échange de séquences de tâches. Une liste tabou conserve un certain nombre d'itérations dans lequel un mouvement d'insertion / échange ne peut pas être inversé. Le processus de diversification est fait à l'aide de la recherche dispersée en utilisant un nombre aleatoire de solutions obtenu de l'ensemble initial. Après un nombre d'itérations, l'opérateur qui a donné les meilleurs résultats est utilisé pour continuer la recherche.

2.2 CARP avec contraintes de périodicité (PCARP)

Le PCARP est une extension du CARP auquel on ajoute des contraintes de périodicité des visites, c'est-à-dire, les arcs d'un réseau peuvent être visités plus d'une fois dans un horizon de temps. La demande de chaque arc dans le réseau est différente. Quelques arcs ont une demande très grande, d'autres ont une demande très petite. La solution est de générer un horizon qui permet de laisser de côté les arcs à faible demande pour une ou plusieurs périodes et se concentrer sur les arcs à forte demande. Cependant, à la fin de l'horizon chaque arc sera servi au moins une fois. Certains seront servis plus d'une fois.

Afin d'illustrer l'importance du PCARP et sa différence avec le CARP, on va utiliser l'application de la collecte de déchets. La figure 2.2 montre deux trajets qui peuvent être réalisés par un ou deux véhicules. Certains arcs (ceux en plus foncé) ont besoin de service. La figure 2.3 montre les mêmes trajets répétés pour un horizon de temps d'une semaine. Si on applique une approche CARP, la même solution est utilisée chaque jour (période). Cependant certains arcs ne génèrent pas une quantité suffisant de déchets pour être servis et on peut seulement les traverser. L'approche PCARP comprend une solution pour la semaine complète plutôt que pour un seul jour, de telle façon qu'on modifie les trajets pour accommoder seulement les arcs qui ont besoin de service.

Le PCARP a été introduit par Lacomme *et al.* (2002). Ils ont ajouté l'élément de périodicité

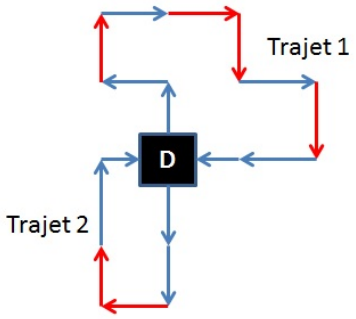

Figure 2.2 Deux trajets pour le CARP.

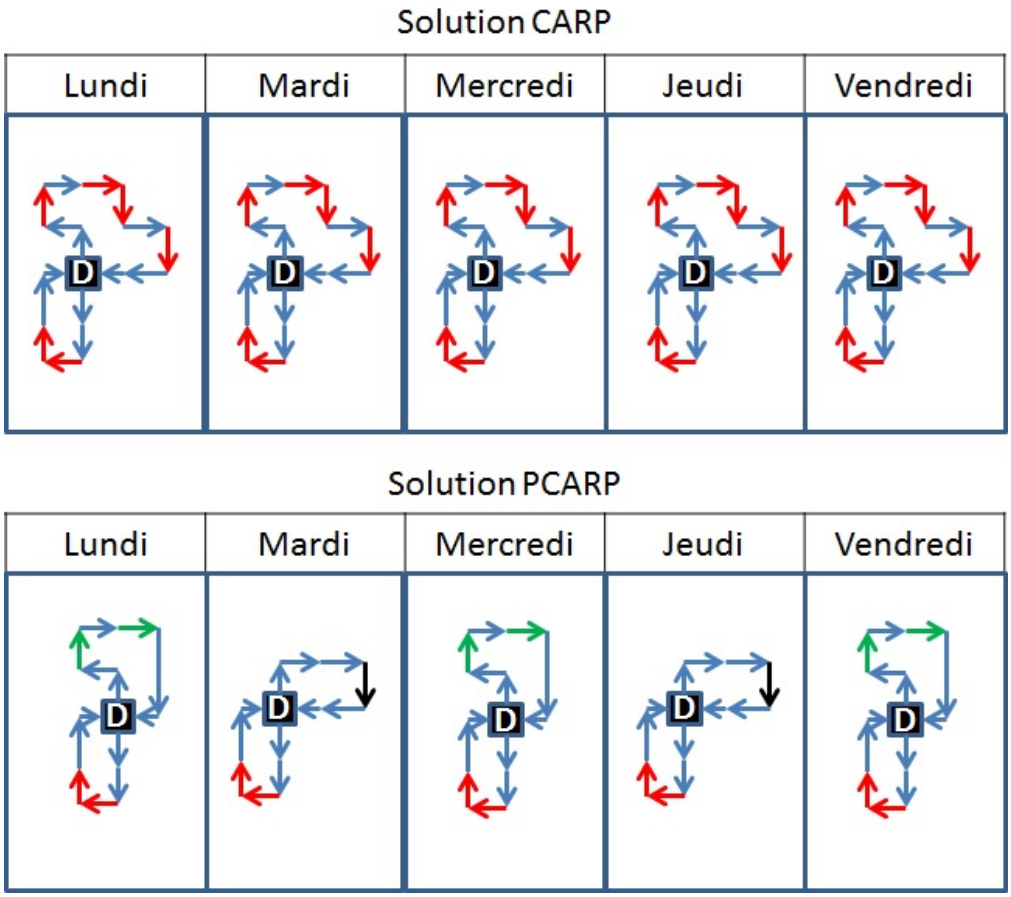

Figure 2.3 Différence entre utiliser une solution pour un jour et une solution pour une semaine complète.

aux problèmes de tournées sur les arcs avec capacités pour déterminer la tournée des véhicules à chaque jour plutôt que d'avoir une solution générale donnée par le CARP. De plus, les applications du CARP trouvées dans la littérature considéraient des réseaux non orientés

et un seul horizon de temps (Lacomme *et al.*, 2005). Toutefois les applications réelles comme la collecte des déchets se font généralement sur un réseau mixte.

Il est important d'établir les caractéristiques générales du PCARP décrites dans Lacomme *et al.* (2002), Chu *et al.* (2005) et Lacomme *et al.* (2005).

- Le PCARP est défini sur un horizon de temps H. Cet horizon peut changer d'une application à l'autre mais est toujours composé de périodes de temps. On a un total de P périodes de temps de même durée.
- On a un ensemble de V véhicules, chacun avec une capacité Q. Dans les applications trouvées, tous les véhicules ont la même capacité. Néanmoins, il est possible d'avoir une flotte de véhicules avec des capacités différentes.
- Les véhicules doivent débuter la tournée au dépôt initial et finir au même dépôt ou à un autre.
- Le réseau $G = (N, E \cup A)$ est défini par N, l'ensemble des intersections de rues ; E, l'ensemble des segments de route non orientés et A, l'ensemble de segments de route orientés. C'est ainsi que le graphe devient mixte.
- Les arêtes et les arcs ont un coût de traversée C_{ij}, une demande q_{ij} et une activité avec une fréquence f_{ij} qui représente le nombre de services que le segment (i,j) nécessite dans l'horizon T. La fréquence est bornée par le nombre des périodes, ainsi, $1 \leq f_{ij} \leq P$ mais au plus une fois par période.
- On parle de périodicité lorsque les activités de service se déroulent à intervalles réguliers.
- On appelle espacement le temps écoulé entre chaque visite.

Le PCARP est NP-dur (Lacomme *et al.*, 2002) parce qu'il inclut le cas principal du CARP avec une seule période. Golden et Wong (1981) ont démontré que le CARP était NP-dur.

2.2.1 Modèles de programmation linéaire pour le PCARP

Modèle de combinaisons jour-tâche

Le premier modèle de programmation linéaire en nombres entiers pour le PCARP a été proposé par (Chu *et al.*, 2005). Quelques considérations importantes pour ce modèle :

- Le modèle est basé sur un graphe non orienté $G = (X, E)$ où X représente l'ensemble de nœuds et E l'ensemble d'arêtes.
- Un ensemble $R \subseteq E$ représente les tâches (arêtes à servir). Chaque tâche a une fréquence de réalisation f_{ij}. Un sous-ensemble $E_R(S) \subseteq E$, où $S \subseteq X$ représente les arêtes qui

sont coupées par la coupe (S,X-S).
- Tous les véhicules sont identiques et ils ont la même capacité W.
- Il existe un ensemble de combinaisons possibles K des périodes $p = 1, 2, \ldots, P$. Le nombre de périodes dans chaque combinaison est relié à la fréquence f_{ij}.

On a une matrice A (0 ou 1) de taille $K \times P$, où K est le nombre de combinaisons possibles et P est le nombre de périodes sur un horizon T. La matrice A s'élabore avant de résoudre le modèle en considérant les bornes supérieure et inférieure d'espacement entre deux visites sur la même arête. Par exemple, si la fréquence pour une certaine arête est $f_{ij} = 2$, et l'horizon est d'une semaine, une combinaison peut être (lundi, jeudi) ou (mardi, vendredi). Un exemple de combinaisons possibles est montré au tableau 2.1 pour des combinaisons de 2 et 3 visites par semaine.

Tableau 2.1 Exemple de combinaisons pour une fréquence de 2 et 3 jours par semaine.

	Période				
$f(u) = 2$					
K / p	Lundi	Mardi	Mercredi	Jeudi	Vendredi
1	x				x
2	x			x	
3	x		x		
4		x		x	
5			x		x
...					
$f(u) = 3$					
1	x		x		x
2		x	x	x	
...					

Modèle M1 (Chu *et al.*, 2005).

Soit la variable $x_{ijvp} = 1$ si l'arête (i, j) est traversée du sommet i vers j par le véhicule v durant la période p, 0 sinon ; $l_{ijvkp} = 1$ si la tâche (i, j) s'effectue du sommet i vers j par le véhicule v, à la période p en utilisant la combinaison k ; 0 sinon ; et $z_{ijk} = 1$ si la tâche (i, j) utilise la combinaison k ; 0 sinon. Soit aussi le paramètre C_{ij}, le coût de traverser l'arête (i, j), Q_{ijkp}, la demande de l'arête (i, j) pour la combinaison k durant la période p, et W, la capacité des véhicules. On suppose que tous les véhicules ont la même capacité. On pose $A_{kp} = 1$ si la combinaison k couvre la période p, 0 sinon. La figure 2.4 montre ce modèle.

$$\min \sum_{(i,j)\in E} \sum_{v=1}^{V} \sum_{p=1}^{P} C_{ij}(x_{ijvp} + x_{jivp}) \tag{2.1}$$

s.c :

$$\sum_{k\in comb(i,j)} z_{ijk} = 1 \qquad \forall (i,j) \in R \tag{2.2}$$

$$\sum_{(i,j)\in E} x_{ijvp} = \sum_{(i,j)\in E} x_{jivp} \qquad \forall i \in X, v = 1,\ldots,V, p = 1,\ldots,P \tag{2.3}$$

$$x_{ijvp} \geq \sum_{k\in comb(i,j)} l_{ijvkp} \quad \forall (i,j) \in R, v = 1,\ldots,V, p = 1,\ldots,P \tag{2.4}$$

$$x_{jivp} \geq \sum_{k\in comb(i,j)} l_{jivkp} \quad \forall (i,j) \in R, v = 1,\ldots,V, p = 1,\ldots,P \tag{2.5}$$

$$\sum_{v=1}^{V} (l_{ijvkp} + l_{jivkp}) = A_{kp} Z_{ijk} \qquad \forall (i,j) \in R, k \in comb(i,j), p = 1,\ldots,P \tag{2.6}$$

$$\sum_{(i,j)\in R} \sum_{k\in comb(i,j)} Q_{ijkp}(l_{ijvkp} + l_{jivkp}) \leq W \qquad \forall p = 1,\ldots,P, v = 1,\ldots,V \tag{2.7}$$

$$\sum_{i\in S} \sum_{j\notin S} x_{ijvp} \geq (l_{ijvkp} + l_{jivkp}) \quad \forall S \subseteq \{2,\ldots,n\}, (r,s) \in E_R(S), \; p = 1,\ldots,P, k \in comb(i,j), v = 1,\ldots,V \tag{2.8}$$

$$x_{ijvp}, x_{jivp}, z_{ijk}, l_{ijvkp}, l_{jivkp} \in \{0,1\} \qquad \forall (i,j) \in R, p = 1,\ldots,P, v = 1,\ldots,V \tag{2.9}$$

Figure 2.4 Modèle M1 pour le PCARP.

La fonction objectif (2.1) permet de minimiser le coût C_{ij} de traverser le réseau en trouvant une combinaison de périodes pour executer chaque tâche f_{ij} fois dans l'horizon de temps. Les contraintes (2.2) permettent qu'une seule combinaison k soit affectée à chaque tâche (i,j). Les contraintes (2.3) sont les contraintes de conservation de flux. Les contraintes (2.4) et (2.5) permettent d'assurer qu'on puisse servir l'arête (i,j) si et seulement si on la traverse. Dans les contraintes (2.6), une arête est servie durant la période p si et seulement si p fait partie de la combinaison k qui a été sélectionnée pour la tâche (i,j). Ces contraintes permettent

aussi que l'arête (i, j) soit servie par le véhicule v dans une seule direction. Les contraintes (2.7) permettent de ne pas dépasser la capacité des véhicules. Dans les contraintes (2.8) si on traverse une arête de l'ensemble $E_R(S)$, il faut traverser la coupe $(S, X - S)$. Cette contrainte permet d'éviter la formation de cycles sur des arêtes non servies. Finalement, les contraintes (2.9) sont les contraintes de non-negativité.

L'importance de ce modèle, à part d'être le premier modèle proposé pour le PCARP, est qu'il introduit un schéma de combinaisons qui facilite l'utilisation d'algorithmes heuristiques pour des problèmes de grande taille. Néanmoins, le nombre de combinaisons augmente lorsque l'horizon de temps est plus grand, ce qui rend cette méthode inappropriée pour les exemplaires qui supposent un horizon de temps qui tend vers l'infini.

Ce modèle a été utilisé pour résoudre des exemplaires de problèmes de petite taille : 5 périodes et 10 activités sur les arêtes. C'est la raison pour laquelle les auteurs ont proposé une méthode heuristique pour résoudre le problème.

Modèle pour le PCARP avec services irréguliers

Un modèle pour le PCARP avec services irréguliers a été proposé par Monroy *et al.* (2011). Les hypothèses faites pour ce modèle sont :

- On travaille sur un réseau orienté $G = (N, A)$.
- Les arcs sont divisés en C classes selon leur fréquence de service. L'ensemble Ω_c est l'ensemble d'arcs appartenant à la classe C. T est l'union des ensembles Ω_c.
- Il existe un horizon de temps H divisé en l périodes de temps (jours). Un ensemble e_c contient des regroupements possibles de périodes qui sont appelées *sous-périodes*. Le tableau 2.2 montre un exemple d'un horizon de temps divisé en cinq jours (lundi - vendredi) et cinq sous-périodes qui regroupent un nombre différent de jours.
- Il existe un ensemble d'arcs $S \subseteq A$ et de nœuds $N(S) \subseteq N$ incidents aux arcs de S.
- L'ensemble $O(n)$ contient les arcs sortants du sommet $n \in N$. L'ensemble $I(n)$ contient les arcs entrents au sommet $n \in N$.
- L'ensemble $R \subseteq A$ est l'ensemble des arcs qui ont besoin de service.
- Les véhicules travaillent deux quarts par jour. Pour chaque quart, il faut déterminer une route de service par véhicule. L'ensemble V est l'ensemble de routes possibles.

Dans cette application, la fréquence des services est comptée par sous-période plutôt que par période sur l'horizon H. Ces sous-périodes sont définies à l'avance et peuvent avoir plus d'une période. Il s'agit d'une stratégie similaire à l'utilisation de combinaissons du modèle

précédent.

Tableau 2.2 Exemple de définition de sous-périodes.

Sous-période ($j \in e_c$)	Période
1	lundi
2	lundi, mardi
3	Jeudi
4	Mardi, Mercredi, Jeudi
5	Jeudi, Vendredi

Ensuite, on définit la fréquence des services f_j pour chaque sous-période. Les avantages de cette méthode sont les mêmes que ceux d'avoir un ensemble de combinaisons, c'est-à-dire on travaille avec un nombre réduit d'éléments à combiner.

Cette approche marche bien pour des horizons de temps relativement petits comme une semaine où le nombre de combinaisons n'est pas trop grand. Un grand nombre de combinaisons augmentera le nombre de variables dans les modèles mathématiques.

Considérons les variables $x_a^k = 1$ si l'arc $a \in R$ est servi par la route k, 0 sinon, et y_a^k qui est le nombre de fois qu'on traverse, sans service, l'arc a en utilisant la route k. Le paramètre P_a est la priorité de l'arc a ; f_j est la fréquence de service pour la sous-période j ; w_j est l'ensemble de routes pour la sous-période j ; M est une très grande constante dont la valeur est au moins égale au nombre maximum de fois qu'on traverse un arc ; c_a est le coût de service de l'arc a et t_a est le coût de traverser l'arc a sans service ; Q est la capacité des véhicules. Dans ce problème, la capacité est mesurée en termes de la longueur ou temps limite pour chaque véhicule. Les coûts c_a et t_a ont les mêmes unités de mesure. La figure 2.5 montre le modèle **M2**.

La fonction objectif (2.10) maximise la priorité. Les contraintes (2.11) sont les contraintes de conservation de flux. Les contraintes (2.12) permettent que la capacité (temps ou longueur) des véhicules soit respectée. Les contraintes (2.13) assurent la connectivité du réseau. Les contraintes (2.14) permettent que la fréquence soit respectée pour chaque sous-période et pour chaque type d'arc. Les contraintes (2.15) et (2.16) definissent les variables du problème.

Ce modèle est un exemple d'utilisation du coût et du temps comme contraintes de capacité, plutôt que la traditionnelle capacité physique des véhicules, ce qui permet plus de versatilité

$$\max \sum_{k \in V} \sum_{a \in R} P_a x_a^k \tag{2.10}$$

s.c :

$$\sum_{a \in O(n)} (x_a^k + y_a^k) - \sum_{a \in l(n)} (x_a^k + y_a^k) = 0 \qquad \forall k \in V, n \in N \tag{2.11}$$

$$\sum_{a \in R} (c_a x_a^k) + \sum_{a \in A} (t_a y_a^k) \leq Q \qquad \forall k \in V \tag{2.12}$$

$$M \sum_{a \in l(N(S))} (x_a^k + y_a^k) \geq \sum_{a \in O(N(S))} (x_a^k + y_a^k) \quad \forall S \subset A, l \notin N(S), k \in V \tag{2.13}$$

$$\sum_{k \in w_j} x_a^k \geq f_j \qquad \forall j \in e_c, a \in \Omega_c, c = 1, \ldots, |T| \tag{2.14}$$

$$x_a^k \in \{0, 1\} \qquad \forall k \in V, a \in R \tag{2.15}$$

$$y_a^k \in \mathbb{Z}^+ \qquad \forall k \in V, a \in A \tag{2.16}$$

Figure 2.5 Modèle M2

dans la fonction objectif.

Modèle mathématique pour minimiser les coûts d'investissement

Le modèle proposé par Mei *et al.* (2011) vise à minimiser principalement les coûts d'investissement générés par l'achat de véhicules. Leur hypothèse est que ces coûts sont significativement supérieurs aux coûts de routage, alors, s'ils ont priorité on peut réduire le coût total.

$$\min f(S) = \alpha \cdot mnv(S) + tc(S) \tag{2.17}$$

où S est une solution du problème, $mnv(S)$ est le nombre de véhicules utilisés dans la solution S, $tc(S)$ est le coût total de routage de la solution S. Le paramètre α est une constante suffisamment grande pour donner la priorité au premier objectif.

Ce modèle est très général, alors, les contraintes expriment les caractéristiques d'un problème périodique.

- Chaque trajet débute et finit au dépôt.
- Un segment de route ne peut pas être servi plus d'une fois dans un période de temps.

- On peut utiliser seulement les combinaisons des périodes existantes. Ces combinaisons ont été définies précédemment, de la même façon que dans Chu *et al.* (2005).
- La somme des demandes des arcs servis ne peut pas dépasser la capacité du véhicule.
- La fréquence de service de chaque arc doit être respectée.

Les modèles mentionnés sont accompagnés par des méthodes heuristiques ou métaheuristiques utilisées pour la résolution des exemplaires réels. Ces méthodes seront décrites dans la section suivante.

2.2.2 Heuristiques et métaheuristiques pour le PCARP

Les modèles mathématiques décrits dans la section précédente fonctionnent pour des exemplaires de petite taille. Dans la plupart des applications, la taille des modèles augmente exponentiellement avec la taille du problème (nombre de nœuds et d'arcs). On a besoin d'utiliser des méthodes heuristiques ou métaheuristiques pour résoudre les instances de grande taille qui se trouvent dans la réalité.

Un algorithme mémétique a été presenté dans Lacomme *et al.* (2002) et plus tard detaillé dans Lacomme *et al.* (2005) pour traiter le problème de collecte des déchets dans la ville de Troyes, en France. Les algorithmes mémétiques sont une forme d'algorithmes évolutifs qui combinent des algorithmes génétiques avec une recherche locale dans le processus de mutation. Cette algorithme utilise les élements similaires que celui proposé par Lacomme *et al.* (2001) pour le CARP. L'objectif est de réduire le coût total des routes sur l'horizon de temps. Les combinaisons de jours pour chaque tâche sont fixées selon leur fréquence. Cela permet d'utiliser l'algorithme *Route first-cluster second* d'Ulusoy pour évaluer une séquence de tâches par période (jour). On trouve la meilleure combinaison pour chacun des jours et finalement on réunit les solutions trouvées pour chaque période de l'horizon pour avoir la solution du problème. La population initiale de chromosomes est générée aléatoirement à l'exception d'une solution obtenue en utilisant une heuristique d'insertion. Le croisement est fait de façon à respecter les combinaisons tâche/jour établies. On copie une séquence des tâches d'un parent. Celles-à doivent être placées à la même période dans la nouvelle solution si ce placement est en accord avec les combinaissons tâche/jour. De la même façon, le reste des tâches, hors de la séquence choisie, sont copiées d'un deuxième parent dans l'ordre et en respectant les combinaisons tâche/jour. Le processus de mutation est fait par une recherche locale pour chaque période. En d'autres mots, on change l'ordre des tâches durant une journée mais on ne les change pas d'un jour à l'autre. Ces séquences de tâches sont évaluées pour

trouver un nouveau chromosome.

Cette modification au processus de croisement nous indique la complexité du PCARP par rapport au CARP. Le nombre d'enfants qu'on est capable de créer est limité par le nombre de combinaisons tâche/jour disponibles.

Les tests ont été faits pour deux ensembles d'exemplaires utilisés antérieurement pour le CARP et modifiés pour inclure un horizon de temps. Ils ont été réalisés pour quatre heuristiques qui diffèrent par leur nombre d'éléments correspondant à l'algorithme évolutif. Les résultats montrent que l'algorithme avec le plus d'éléments donne les meilleurs résultats en général.

En plus d'élaborer un modèle mathématique pour le PCARP, Chu *et al.* (2005) ont proposé trois algorithmes pour résoudre ce problème : deux algorithmes heuristiques d'insertion et un algorithme heuristique à deux phases. Les trois algorithmes prennent en compte les combinaisons tâche-jours décrites pour le modèle **M1**. Les méthodes d'insertion visent à insérer les tâches dans un trajet réalisable pour chaque jour de chaque combinaison. Le coût total d'insertion pour chaque combinaison est ensuite calculé et la combinaison avec le coût le plus bas est sélectionnée. Dans la méthode à deux phases, on sélectionne un ensemble de tâches pour chaque jour en utilisant une borne inférieure dévéloppée pour le CARP, puis on sélectionne la combinaison tâche-jour avec le coût le plus bas.

Le problème du postier rural périodique a été traité dans Ghiani *et al.* (2005) pour la collecte des déchets en appliquant une heuristique qui minimise le dead-heading. Avec une fréquence de collecte donnée, l'algorithme sélectionne une combinaison de tâches à servir chaque jour et applique une recherche locale pour trouver la moins chère. Puis, on modifie les combinaisons. Pour évaluer l'algorithme, ils ont calculé la solution optimale pour une petite instance du problème et ont comparé celle-ci à la valeur obtenue par l'heuristique. Cet algorithme a obtenu des solutions dont la valeur se situe entre 0.58% et 2.92% de la solution optimale pour différentes distributions du nombre des arêtes obligatoires du graphe.

Un algorithme de recherche dispersée a été proposé dans Chu *et al.* (2006). La recherche dispersée est une méthode de population avec un processus de diversification qui explore d'autres régions de l'espace des solutions (Martí *et al.*, 2006).

Dans ce problème, on cherche à minimiser le nombre de véhicules et le coût total des routes.

Après avoir trouvé une combinaison avec le nombre minimum de véhicules, ce nombre est fixé et on modifie la solution pour trouver le coût minimum. Les auteurs appliquent les techniques de génération de population et d'évaluation des solutions similaires à celles de Lacomme *et al.* (2005). Pour tester cet algorithme, les auteurs ont déterminé une borne inférieure pour chacun des objectifs. Ils utilisent des bornes inférieures du CARP dans un graphe transformé pour obtenir les bornes pour le PCARP (Chu *et al.*, 2003). Les résultats de l'algorithme de recherche dispersée ont été comparés avec ceux obtenus avec un algorithme d'insertion (BIH). L'algorithme de recherche dispersée a réduit la différence entre la borne inférieure et le BIH de 50% à 7.46% pour le nombre de véhicules et de 42.66% à 8.77% pour le coût total.

Un problème périodique d'arrosage des routes dans les mines à ciel ouvert est étudié dans Li *et al.* (2008). Dans cette application il faut se déplacer sur tous les segments du réseau routier d'une mine et répandre de l'eau pour prévenir que la poussière endommage les camions. Le problème a deux objectifs : minimiser le coût total de routage et trouver un emplacement pour un nouveau dépôt d'eau. Il y a des caractéristiques particulières dans ce cas :

- L'horizon de temps est infini. Les camions d'arrosage travaillent 24 heures par jour.
- Il y a un nombre limité de véhicules.
- Le temps de déplacement n'est pas fixé, ça dépend s'il y a d'autres camions dans le même chemin.
- Il y a plus d'un dépôt dans le réseau.

Les auteurs ont développé deux algorithmes pour résoudre le problème de minimiser le coût total. Le premier est une heuristique de flux à coût minimum qui travaille avec l'information de chaque camion au moment d'arriver à un nœud. Les données utilisées sont la position, la capacité utilisée et le temps disponible. Ensuite, on calcule la prochaine destination à partir d'un ensemble de trajectoires disponibles. Le deuxième algorithme est fait pour calculer les deux ou trois destinations suivantes, selon les données disponibles. Cet algorithme prend un ensemble de chemins possibles pour couvrir tous les arcs avec les camions disponibles. Les auteurs utilisent la relaxation lagrangienne car le nombre de routes possibles devient très grand. Ils suggèrent utiliser d'autres techniques telles que la génération de colonnes ou la programmation dynamique. Le problème de localisation d'un autre dépôt est résolu en trouvant la solution au premier problème avec le nouveau dépôt dans un endroit différent à chaque fois. Cependant, les résultats montrent qu'il est moins coûteux d'acheter un camion que de placer un nouveau dépôt.

Le problème de surveillance des routes a été traité en premier dans Marzolf *et al.* (2006).

Dans ce problème, un véhicule parcourt un réseau routier pour détecter les accidents afin d'augmenter la sécurité et la fluidité sur les routes. Les accidents se produisent aléatoirement. Les routes sont surveillées avec une certaine fréquence, mais les véhicules de service doivent quitter leur tournée pour aller à l'endroit de l'accident. En dehors de la formulation mathématique du problème de routage sur les arcs, il faut utiliser un système d'information géographique qui permet au véhicule de reprendre la tournée incomplète après avoir répondu à l'accident. La capacité dans cette application est la durée de chaque quart de travail. Les auteurs proposent trois approches différentes pour traiter ce problème. La première approche est un modèle mathématique dont l'objectif est de couvrir tous les arcs pour les quarts disponibles sur une période de deux semaines. La deuxième approche est une adaptation de la première dans laquelle les routes sont planifiées de nouveau après chaque quart afin de prendre en compte les cas où les véhicules doivent quitter leur itinéraire actuel pour porter secours lors d'un accident. La troisième approche intègre les routes qui ne sont pas couvertes dans un quart précédent en tenant compte de l'importance de chaque arc. L'objectif est de minimiser la distance utilisée pour aller aux arcs ajoutés.

Monroy *et al.* (2011) considèrent le problème de surveillance des rues et présentent un modèle mathématique (décrit dans la section précédente) et une méthode heuristique. La méthode proposée par les auteurs est une méthode en deux phases. La première phase est un algorithme *Cluster first-route second*, dans lequel les arcs sont groupés en minimisant la distance entre eux et en s'assurant que le nombre d'arcs dans chaque groupe n'excède pas une certaine fraction de la capacité. Dans la deuxième phase, un modèle mathématique est exécuté pour chaque groupe. L'objectif est de maximiser la priorité de différentes classes d'arcs.

Un algorithme de colonie de fourmis a été dévéloppé dans Kansou et Yassine (2009). Ils utilisent un système de combinaisons de périodes associées à chaque arc similaire à celui de Chu *et al.* (2005). L'objectif est de minimiser la distance totale parcourue. L'heuristique de colonie de fourmis est utilisée pour trouver l'ordre dans lequel les arcs doivent être servis, en fonction de la distance entre eux. Ensuite, une combinaison des périodes est trouvée pour l'arc choisi et on utilise un algorithme d'insertion pour ajouter les arcs suivants dans chaque période de l'horizon de temps.

Mei *et al.* (2011) ont utilisé un algorithme mémétique suivi d'un processus de fusion des routes pour réduire les coûts de routage et d'investissement dans le problème périodique. Deux solutions parents créent une solution fille. Tenant compte que l'objectif principal est de minimiser le nombre de véhicules qui circulent dans le réseau (coût d'investissement), dans

cette nouvelle solution, toutes les paires de trajets sont évaluées en fonction de leur demande combinée. La paire avec la plus petite demande combinée est fusionnée de telle façon qu'on réduise le nombre de véhicules en 1 dans chaque itération. Finalement on termine avec un processus de recherche locale.

2.2.3 Autres applications du PCARP

D'autres applications sont mentionnées dans la littérature, mais il n'y a encore aucune étude de PCARP sur celles-ci. Corberán et Prins (2010) et Lacomme *et al.* (2005) mentionnent des applications comme l'inspection des lignes électriques, la fauchage de l'herbe le long des routes, l'épandage de déglaçant sur les routes en hiver et le traitement des voies ferrées avec des herbicides.

Sbihi et Eglese (2010) font une revue des techniques mathématiques importantes pour la logistique écologique (logistique verte). Ils mentionnent l'importance des techniques correctes de collecte de déchets dans l'utilisation de modèles de CARP et PCARP.

D'autres applications qui ne sont pas mentionnées comme PCARP mais qui appartiennent à ce domaine et pourraient être étudiées dans le cadre d'une approche similaire sont :

- l'entretien des routes en hiver, qui peut aussi être modélisé comme un PCARP. Cependant, ces activités dépendent fortement des conditions météorologiques. Une description des opérations d'entretien des rues se trouve dans Perrier *et al.* (2008).
- l'entretien des routes dans les opérations forestières, comme l'extraction de bois est un autre exemple d'une possible application du PCARP. Dans l'exemple de Martin *et al.* (2000), l'objectif est de trouver une ensemble différent de routes à chaque fois afin de minimiser les dommages causés aux routes par les camions.
- L'épandage de sel en hiver pour éviter les accidents dans les routes en raison de la formation de glace. On utilise un véhicule avec une capacité limitée. La quantité de sel peut varier pour chaque route. Le service doit être fourni à une heure spécifique de la journée (Eglese, 1994). Le temps de début du service determine le coût de service (Tagmouti *et al.*, 2007).

2.3 Discussion

Le problème de recherche qui porte sur l'arrosage de chemins d'une mine à ciel ouvert a été introduit à la section précédente. On a un réseau où les arêtes peuvent être considérées comme

des clients qui ont un niveau de stock représenté par l'humidité nécessaire pour retenir des particules de poussière. Un véhicule part du dépôt et arrose les arêtes du réseau. Le niveau d'humidité diminue et doit être rempli. D'où la nature périodique du problème. L'objectif de ce travail est de proposer une méthode pour trouver un ensemble de routes pour servir les arêtes d'un réseau de façon périodique afin de minimiser les coûts de routage et de pénurie à cause d'un niveau insuffisant d'eau livré. On a établi qu'il s'agit d'un problème périodique de tournées sur les arcs avec contraintes de capacité (PCARP) et on a décrit l'influence que les décisions de routage ont sur le niveau d'inventaire sur chacune des arêtes du réseau.

Dans la revue de littérature, on a mentionné les travaux reliés aux problèmes PCARP et on trouve des éléments à considérer et à changer pour le problème d'arrosage.

Premièrement, les périodes qui forment l'horizon de temps sont des intervalles de la même durée (e.g. jours). Elles sont suffisamment longues pour contenir une tournée complète, laquelle va changer à la période suivante. Pour le problème d'arrosage, il faut considérer des périodes de temps de la même durée, mais suffisamment courtes de telle façon que plusieurs d'entre elles puissent être contenues dans une tournée. Le temps d'arrosage est un élément important dans l'application d'arrosage parce qu'on travaille avec un inventaire qui est consommé en fonction du temps.

Dans les problèmes PCARP décrits, la quantité de matériel sur chaque arête augmente en fonction du temps et elle est enlevée en totalité quand le véhicule traverse l'arête. La figure 2.6a montre ce situation dans laquelle, le matériel est collecté au temps t_1. C'est le cas de l'application à la collecte de déchets. Dans le cas du problème d'arrosage, figure 2.6b, la quantité diminue et elle est réapprovisionnée au moment de la traverse. Cela a un effet dans la quantité restant au véhicule parce qu'il est possible de livrer une petite quantité d'eau sans remplir totalement le niveau d'humidité, mais elle peut diminuer le coût de pénurie.

Il existe un problème similaire dans le domaine de tournées sur les nœuds appelé Inventory routing problem (IRP). Les nœuds du réseau représentent des clients, chacun avec un niveau de stock. Il faut prendre de décisions de routage et de gestion de stocks en même temps (Bertazzi et Speranza, 2012). Ce problème n'a pas été traité dans la littérature dans le domaine des ARP, alors, on a appelé ce problème Periodic Capacitated Arc Routing Problem with Inventory Constraints (PCARP IC).

En regardant les approches utilisées pour traiter les problèmes du PCARP, on peut divi-

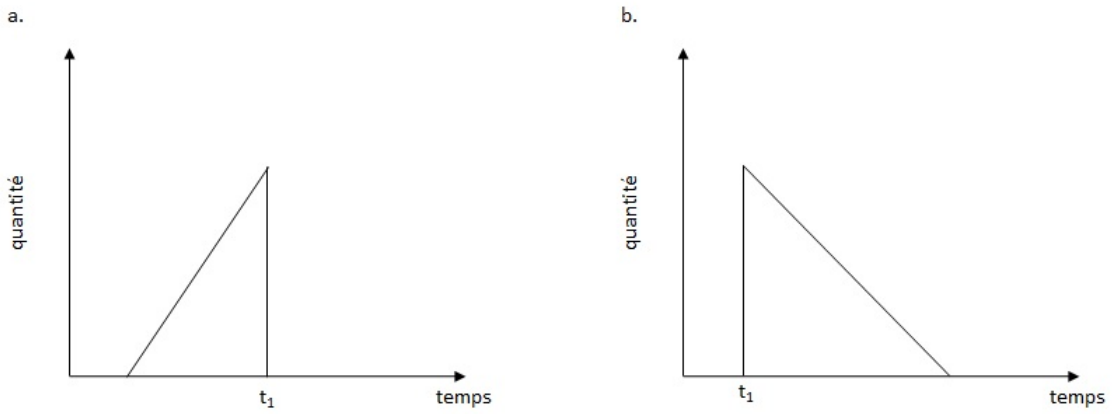

Figure 2.6 a. Augmentation de la quantité de matériel dans les applications du PCARP. b. Réduction de la quantité de matériel dans le problème PCARP avec contraintes de gestion de stocks.

ser l'étude du problème PCARP IC selon les étapes suivantes :

- Trouver un modèle mathématique pour le PCARP IC et déterminer ces limites en termes de la taille du réseau qu'on est capable de résoudre.
- En raison des limites du modèle, trouver un algorithme approprié pour des problèmes de grande taille dans des réseaux réels.
- Continuer l'étude des variations du problème telles que l'utilisation de plusieurs véhicules ou plusieurs dépôts.

Les sections suivantes portent sur la méthodologie à suivre pour l'étude de ce problème.

CHAPITRE 3

ARTICLE 1 : PERIODIC CAPACITATED ARC ROUTING PROBLEM WITH INVENTORY CONSTRAINTS

Cet article (Riquelme Rodríguez *et al.*, 2013)) a été accepté et publié en ligne le 27 novembre 2013 dans *Journal of Operational Research Society*.

Cet article présente l'introduction au problème périodique de tournées sur les arcs avec les contraintes de capacité et de gestion de stocks. Bien que quelques exemples d'applications sont mentionnés, tels que l'arrosage de la végétation le long des rues et l'arrosage des chemins forestiers, l'accent est mis sur l'application d'arrosage des chemins de terre dans les mines à ciel ouvert. Un camion citerne arrose de l'eau sur les routes pour éviter la formation de nuages de poussière. L'humidité des segments de route est consommée par évaporation à cause des conditions météorologiques ou du volume de trafic. La pénurie du niveau d'humidité est pénalisée. Cette pénalité est plus élevée pour les segments de route de plus grande priorité, i.e., les segments de route avec le plus grande volume de trafic. L'objectif est de réduire les coûts de pénalité associés à l'insuffisance d'humidité, tandis que les coûts associés au routage sont aussi réduits.

Les caractéristiques de ce problème qui le rendent unique sont les suivants :

- Les clients sont situés sur les arêtes du réseau et ils ont un niveau de stock qui doit être rempli périodiquement par un véhicule. Les problèmes de routage et de gestion de stocks sont largement étudiées dans un contexte des problèmes de routage sur les nœuds mais aucune application n'a été trouvée dans le domaine de tournées sur les arcs.
- Il y a des décisions concernant le niveau des stocks (niveau d'humidité) qui contribuent à déterminer quels clients desservir. Un camion avec capacité limitée n'est pas en mesure de fournir tous les clients avec le matériel nécessaire.
- Il y a un horizon de temps divisé en périodes ayant la même durée. Les périodes de temps sont les éléments utilisés dans ce modèle pour simuler la traversée, le service et la consommation de la quantité en l'inventaire. Comme cela sera expliqué dans l'article, ils augmentent le nombre de variables, ce qui limite la taille des réseaux qui peuvent être résolus à l'optimalité. À différence des problèmes périodiques de tournées sur les arcs présentés dans la section précédente, dans lesquels une tournée complète est for-

mée pour une période de temps, on utilise plusieurs périodes de temps pour former une tournée.

Les contributions de cet article se résument comme suit :

- On présente le premier modèle mathématique qui combine les décisions sur la gestion des stocks et routage dans le domaine de tournées sur les arcs. Ce modèle est modifié pour inclure deux cas : l'un dans lequel la quantité d'eau à livrer est constante et l'autre dans lequel elle est variable.
- Le modèle mathématique pour l'arrosage des chemins est résolu pour deux ensembles d'exemplaires. La taille des exemplaires est déterminée par le nombre de nœuds et d'arêtes du réseau, ainsi que par le nombre de périodes dans l'horizon de temps qu'on attend à résoudre. On présente les limites de ce modèle en termes de la taille du réseau et des périodes de temps.

Periodic capacitated arc routing problem with inventory constraints

Juan Pablo Riquelme Rodríguez, Michel Gamache, André Langevin
CIRRELT, GERAD and Department of Mathematics and Industrial Engineering
École Polytechnique de Montréal

3.1 Abstract

The purpose of this paper is to study the periodic arc routing problem when the arcs of a network behave as customers, and sufficient material is delivered so that each achieves its desired inventory level. Therefore, routing and inventory decisions are made simultaneously. Applications include dust suppression in open-pit mines or forest roads and plant watering along sidewalks or street medians. A truck periodically sprays water along the edges of a network. The humidity reaches a desired level and is then consumed over time until water is delivered again. The quantity of water delivered can be fixed or variable ; we consider both scenarios and propose a mathematical model for each. Results are reported to validate the model. The contribution of this paper is the first mathematical model that combines inventory and routing decisions in the arc routing domain.
Keywords : Inventory, Linear programming, Periodic arc routing problem, Vehicle routing.

3.2 Introduction

Vehicle routing problems for which the service (e.g., pickup, deliveries) occurs on the arcs are called arc routing problems. A special case is the PCARP in which there is a capacity constraint on the material that can be transported by the vehicle, and some or all of the arcs must be visited more than once in a given time horizon. The problem was introduced by Lacomme, (Lacomme *et al.*, 2002), and it has many applications including garbage collection (Lacomme *et al.*, 2002), (Chu *et al.*, 2005), (Ghiani *et al.*, 2005), road monitoring (Marzolf *et al.*, 2006), and road watering (Li *et al.*, 2008). In garbage collection, for example, arcs accumulate a quantity of material over time, and it is removed when the vehicle arrives. We consider the case where the material flows in the opposite direction. A certain quantity of material is available at the arcs and it is consumed over time until the vehicle arrives and it is replenished. Thus, the quantity of material behaves like the inventory of a customer. Capacity constraints and the frequency of service are considered. Applications include watering roads in open-pit mines and forests and watering plants or gardens in street medians or sidewalks. Water is delivered along the arc to meet a required humidity level. It is then consumed or

evaporated at a certain rate. A water vehicle with a limited capacity services the arcs with varying frequencies within a time horizon.
The contributions of this paper include the mathematical model used to find the optimal route and the quantity to deliver to each arc. To the best of our knowledge, no mathematical model currently exists for this problem in the arc routing domain.
This paper is organized as follows. Section 2 presents a review of PCARP algorithms. The problem is defined in Section 3, and the mathematical model is presented in Section 4. Section 5 describes the tests carried out to validate the model, and Section 6 presents conclusions and future work.

3.3 Literature review

The first periodic application of the capacitated arc routing problem was presented in 2002 (Lacomme *et al.*, 2002). They focus on the garbage collection problem, for which the time horizon of one week is divided into equal-length time periods or days. A set of limited-capacity vehicles start and end their routes at the depot. Each arc on the network has a traversing cost, a demand, and a service frequency that represents the number of services required during the time horizon. The frequency is limited to at most one visit per time period, and the visits are spaced to allow time for the garbage to accumulate. The PCARP is NP-hard (Lacomme *et al.*, 2002) because it includes the Capacitated Arc Routing Problem (CARP) as a special case when there is only one time period.
The first integer linear program for the PCARP was developed by (Chu *et al.*, 2005) for the waste collection problem. It assigns vehicles to routes given a set of feasible day combinations. It is used to solve small instances, and the authors developed three heuristic algorithms to solve larger instances. Lower bounds for the problem were developed by (Chu *et al.*, 2003). (Ghiani *et al.*, 2005) developed a heuristic algorithm for the periodic rural postman problem. It first distributes the visits to streets with the same service frequency and then applies local search to obtain a lower cost solution. Metaheuristic algorithms for the PCARP include a memetic algorithm (Lacomme *et al.*, 2005), a scatter search algorithm (Chu *et al.*, 2006), and an ant colony optimization algorithm combined with an insertion heuristic (Kansou et Yassine, 2009). The road monitoring problem was first treated as a PCARP by (Marzolf *et al.*, 2006). In this problem, a vehicle traverses a road network detecting accidents in order to increase security and traffic flow. Road watering in open-pit mines was studied by (Li *et al.*, 2008). They have two objectives : to minimize the total cost and to find a new location for a water depot. They present two heuristic algorithms that take as input a given position,

the water used, and the time that each truck arrives at a node. Then they select the next destination from a set of available options. (Monroy *et al.*, 2011) developed a mathematical program for the PCARP with irregular services, in which the frequencies for specific tasks vary within the time horizon. (Mei *et al.*, 2011) developed a memetic algorithm with the primary objective of reducing the investment cost rather than the routing cost.

3.4 Problem definition

The problem that we study is road watering in open-pit mines. It was first studied by (Li *et al.*, 2008) for an open-pit copper/gold mine in Australia. They describe the situation as follows :

> In open-pit mines, vehicle travel and action of the wind can result in airborne particulate matters from haul roads that cause fugitive dust emissions. Silicate materials contained in the emissions are harmful to eyes and lungs of mine workers. Dust reduces the visibility of roads ... Dry road surfaces are harmful to the tires of production trucks in the mine. (voir Li *et al.*, 2008, page1)

Because of the temporary nature of mine roads, long-lasting dust suppression methods such as paving or gravel spreading are not appropriate because of the high initial cost and because the roads can easily deteriorate when used by hauling vehicles (Neulicht et Shular, 1998). Thus, water trucks traverse the road network spraying water to suppress dust. Roads are classified according to their priority (Li *et al.*, 2008) ; the most traveled roads have higher priorities and are watered (or serviced) more frequently. A water truck has a limited capacity and when it is empty it returns to the refilling station (depot). To avoid road degradation, there is an upper bound on the water that can be sprayed. The time horizon is infinite since the truck runs twenty-four hours per day.

We propose a different approach to this problem from that suggested in (Li *et al.*, 2008). We use a mixed integer programming model to obtain the optimal truck route and the quantity of water to be sprayed on each road according to its priority. We make the following assumptions :

- The road network has one depot and there is one water vehicle traversing the network. The problem can be extended to more than one vehicle. However, this scenario is outside the scope of this paper. The truck is not able to service all the roads in a single pass so a selection must be made.
- The vehicle has a constant speed. In reality, this speed is affected by several factors, including the presence of other trucks on the road and the fact that watering and not watering require different speeds. We suggest a modification to the model to consider

the case with different watering and non-watering speeds, but our model considers a constant vehicle speed.

The loss of humidity on each road is treated via an inventory consumption function. There are two main factors affecting this : the traffic volume and the water evaporation. The loss of humidity due to traffic volume depends on the importance of the road. The traffic intensity is greater on main roads, which therefore require a greater watering frequency. Water evaporation, on the other hand, depends on the time of the day. For non-paved roads it is estimated using class A pan evaporation (Neulicht et Shular, 1998). A pan is filled with water to a certain level and after twenty-four hours the new level is measured. The difference observed is the evaporation for one day. In this paper, we use the results for an hourly class A pan evaporation simulation found by (Molina-Martínez *et al.*, 2006). According to them, the evaporation reaches its peak between noon and 16 :00 and is of course minimal at night. The water truck consists of a tank, a pump, the piping, and the nozzles located at the rear. The operator can turn the pump on or off to control the spraying. Some trucks allow the operator to activate individual water sprays to control the amount of water (Cecala *et al.*, 2012). We consider two cases. The first is the more general case in which the amount of water can be controlled, and we have to decide which roads to visit and how much water to spray on each. The second case assumes a fixed rate of spraying.

3.5 Mixed integer programming model

This section presents a mathematical model for road watering in open-pit mines. The roads form an undirected network $G = (N, E)$ in which the edges can be traversed in any direction. We consider that $(i, j) \in E$ implies that $i < j$. For our model, this network is transformed into a directed network $G = (N, A)$, where N is the set of nodes and A is the set of arcs, and in which for each edge (i, j) there are two arcs, (i, j) and (j, i). We include an artificial arc at the depot, the loop $(0, 0)$, whose purpose is to allow the vehicle to refill when it returns to the depot. The set A^+ includes the arcs in A and the artificial arc, i.e., $A^+ = A \cup \{(0, 0)\}$. The set Ω includes the arcs whose length is greater than 1, i.e., $\Omega = \{(i, j) \in A | d_{ij} > 1\}$. The use of this set will be explained later. Figure 3.1a) shows an example of the road network in an open-pit mine. The number next to each edge represents its length in distance units. These units are proportional to the real length of the road. Figure 3.1b) shows the transformation to a directed network and the artificial arc $(0, 0)$.

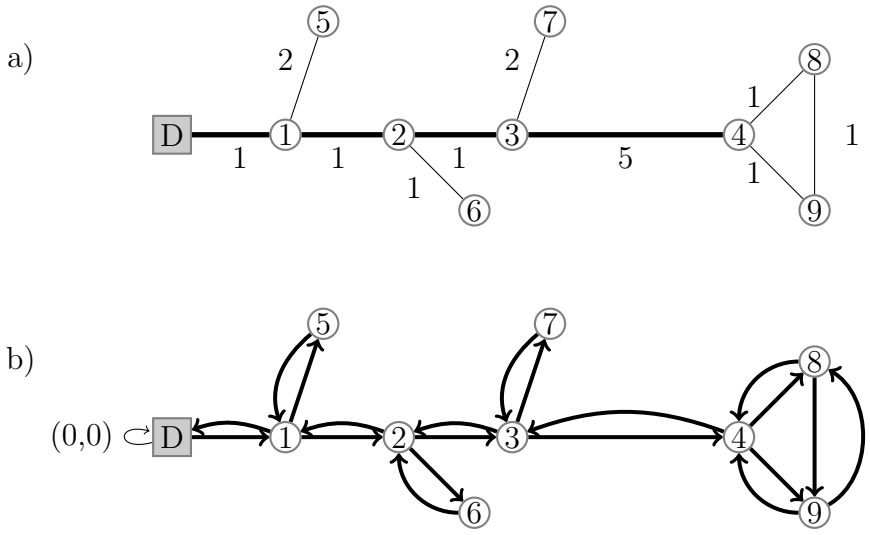

Figure 3.1 *a)* Representation of the road network in an open-pit mine. *b)* Directed network with the artificial arc $(0, 0)$ at the depot.

3.5.1 Inventory cost

Each edge $(i, j) \in E$ has an initial level of humidity that is gradually consumed at a given rate during a time horizon from T_0 to T. For edge (i, j), let H_{ij}^t be the level of humidity at time t, where $T_0 \leq t \leq T$, and $\overline{h_{ij}}$ the desired level of humidity to ensure particle retention. We also define a maximum humidity level H_{ij}^{max} to prevent over-watering and an initial humidity level L_{ij}. Figure 3.2 shows the inventory level for edge (i, j) when a quantity q_{ij} of water is supplied at time T_u. Figure 3.3 shows an approximation of the inventory if the time is discretized into the periods $t \in \{T_0,\ T_0 + \delta,\ T_0 + 2\delta, \ldots, T - \delta,\ T\}$. Regardless of the length of the edge, we consider that all the quantity is delivered at the beginning of the service. The quantity supplied at the beginning of each time period is represented by q_{ij}^t. As explained before, two factors contribute to humidity consumption. The percentage of evaporation caused by solar radiation depends on the time of the day and is represented by e_t, while the percentage of consumption caused by traffic volume depends on the type of edge. It is represented by f_{ij}. The total consumption is $e_t + f_{ij}$.

The area under the curve $\overline{h_{ij}}$ for an edge $(i, j) \in E$ is :

$$Area = \sum_{T_0}^{T} \max\{0, \overline{h_{ij}} - H_{ij}^t\}.$$

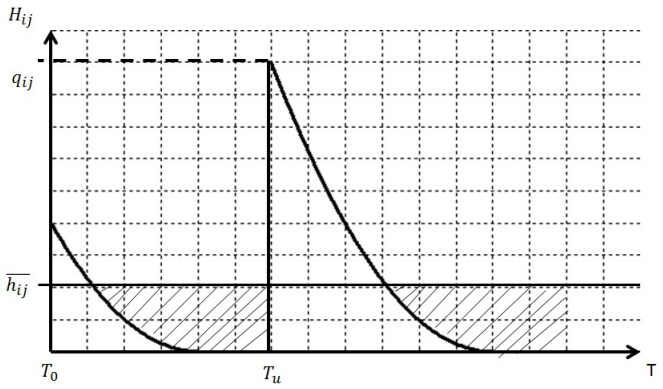

Figure 3.2 Humidity inventory of edge (i, j).

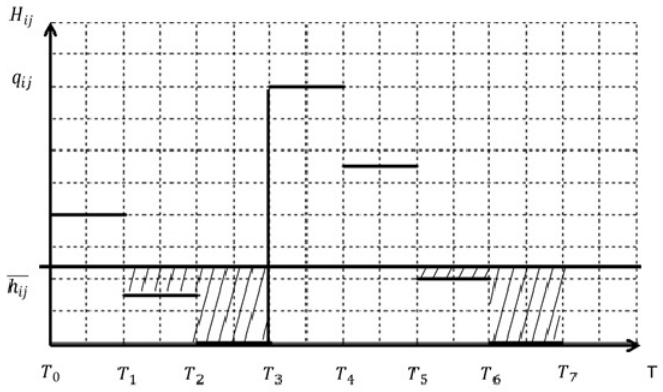

Figure 3.3 Discretization of time.

The area represents the number of liters per meter needed for a certain efficiency of dust particle retention on the ground. There is a penalty P_{ij} if the humidity of edge (i, j) is below the desired level, i.e., if $H_{ij}^t < \overline{h_{ij}}$. The objective is to minimize the product of the penalty and the area :

$$\min \sum_{(i,j)\in E} \sum_{T_0}^{T} P_{ij} \max\{0, \overline{h_{ij}} - H_{ij}^t\}.$$

Variable w_{ij}^t represents the area to be penalized, which is 0 if $H_{ij}^t \geq \overline{h_{ij}}$ and $\overline{h_{ij}} - H_{ij}^t$ otherwise. Substituting $w_{ij}^t = \max\{0, \overline{h_{ij}} - H_{ij}^t\}\ \forall (i, j) \in E$ we have

$$\min \sum_{(i,j)\in E} \sum_{T_0}^{T} P_{ij} w_{ij}^t. \tag{3.1}$$

The constraints associated with Eq. (3.1) are $w_{ij}^t \geq 0$ and $w_{ij}^t \geq \overline{h_{ij}} - H_{ij}^t$ for all edges $(i,j) \in E$ and all time periods $t \in \{T_0, \ldots, T\}$.
We define a maximum vehicle capacity Q^{max} that limits the quantity of water delivered. The variable Q^t represents the water level of the truck at time t.

3.5.2 Routing cost

For each arc $(i,j) \in A$, consider two binary variables : $Y_{ij}^t = 1$ if the vehicle starts traversing arc (i,j) at the beginning of time t without watering it, and 0 otherwise ; and $X_{ij}^t = 1$ if the vehicle starts watering arc (i,j) at the beginning of time t, and 0 otherwise. There are two costs associated with each arc : c_{ij} is the cost of watering and r_{ij} is the cost of traversing the arc without watering it. The parameter c_{00} is the cost of refilling the vehicle. The routing objective is to minimize both costs :

$$\min \sum_{(i,j)\in A^+} \sum_{T_0}^{T} (c_{ij} X_{ij}^t + r_{ij} Y_{ij}^t). \tag{3.2}$$

The penalty costs should dominate the routing costs, so the parameter P_{ij} is at least the maximum traversing or watering cost.

3.5.3 Mathematical model

The following list summarizes the variables used in the model :

q_{ij}^t	Quantity of water used in edge $(i,j) \in E$ at the beginning of time t.
Q^t	Quantity of water in the vehicle at the beginning of time t.
H_{ij}^t	Humidity level in edge $(i,j) \in E$ at the beginning of time t.
$X_{ij}^t = 1$	if arc $(i,j) \in A^+$ is watered at the beginning of time t, 0 otherwise.
$Y_{ij}^t = 1$	if arc $(i,j) \in A^+$ is traversed without watering at the beginning of time t, 0 otherwise.
w_{ij}^t	$\max\{0, \overline{h_{ij}} - H_{ij}^t\}$, $\forall (i,j) \in E$.

The following list summarizes the parameters used in the model :

H_{ij}^{max}	Maximum level of humidity allowed in edge $(i,j) \in E$.
$\overline{h_{ij}}$	Humidity level in edge $(i,j) \in E$ required to achieve certain percentage of particle retention.
Q^{max}	Maximum capacity of the water vehicle.
d_{ij}	Number of time units required to traverse the arc $(i,j) \in A^+$.
L_{ij}	Initial level of humidity in edge $(i,j) \in E$.
c_{ij}	Watering (service) cost of arc $(i,j) \in A^+$.
r_{ij}	Traversing (without service) cost of arc $(i,j) \in A^+$.
P_{ij}	Penalty cost for not watering edge $(i,j) \in E$.
e_t	Percentage of humidity lost by solar radiation.
f_{ij}	Percentage of humidity lost by traffic volume.

The mathematical model is :

$$\min Z = \sum_{(i,j)\in E} \sum_{T_0}^{T} P_{ij} w_{ij}^t + \sum_{(i,j)\in A^+} \sum_{T_0}^{T} (c_{ij} X_{ij}^t + r_{ij} Y_{ij}^t) \tag{3.3}$$

s.t. :

$$w_{ij}^t \geq \overline{h_{ij}} - H_{ij}^t \qquad \forall (i,j) \in E,\ t \in \{T_0, \ldots, T\} \tag{3.4}$$

$$H_{ij}^t = (1 - (e_t + f_{ij}))H_{ij}^{t-1} + q_{ij}^t \qquad \forall (i,j) \in E,\ t \in \{T_0, \ldots, T\} \tag{3.5}$$

$$H_{ij}^0 = L_{ij} \qquad \forall (i,j) \in E \tag{3.6}$$

$$q_{ij}^t \leq H_{ij}^{max}(X_{ij}^t + X_{ji}^t) \qquad \forall (i,j) \in E,\ t \in \{T_0, \ldots, T\} \tag{3.7}$$

$$Hij^t \leq H_{ij}^{max} \qquad \forall (i,j) \in E,\ t \in \{T_0, \ldots, T\} \tag{3.8}$$

$$\sum_{(i,j)\in E} q_{ij}^t \leq Q^t \qquad \forall t \in \{T_0, \ldots, T\} \tag{3.9}$$

$$Q^0 = Q^{max} \tag{3.10}$$

$$Q^t \leq Q^{max} \qquad \forall t \in \{T_0, \ldots, T\} \tag{3.11}$$

$$Q^{t+1} = Q^t - \sum_{(i,j)\in E} q_{ij}^t - q_{00}^t \qquad \forall t \in \{T_0, \ldots, T-1\} \tag{3.12}$$

$$q_{00}^t \geq Q^t - Q^{max} \qquad \forall t \in \{T_0, \ldots, T\} \tag{3.13}$$

$$q_{00}^t \geq -Q^{max} X_{00}^t \qquad \forall t \in \{T_0, \ldots, T\} \tag{3.14}$$

$$X_{ij}^t + Y_{ij}^t \leq 1 \qquad \forall (i,j) \in A^+,\ t \in \{T_0, \ldots, T\} \tag{3.15}$$

$$\sum_{(i,j)\in A|i=0} X_{ij}^0 + Y_{ij}^0 = 1 \tag{3.16}$$

$$\sum_{(i,j)\in A|j=0} X_{ij}^T + Y_{ij}^T = 1 \tag{3.17}$$

$$X_{ij}^t + Y_{ij}^t \leq \sum_{k|(j,k)\in A^+} X_{jk}^{t+d_{ij}} + Y_{jk}^{t+d_{ij}} \qquad \forall (i,j) \in A^+,\ t \in \{T_0, \ldots, T\} \tag{3.18}$$

$$X_{ij}^t + Y_{ij}^t \leq S_{ij}^{t+m} \qquad \forall (i,j) \in \Omega,\ m \in \{0, \ldots, d_{ij} - 1\},\ t \in \{T_0, \ldots, T-m\} \tag{3.19}$$

$$\sum_{(i,j)\in A\setminus\{(u,v)\}} (X_{ij}^t + Y_{ij}^t) \leq 1 - S_{uv}^t \qquad \forall t \in \{T_0, \ldots, T\},\ (u,v) \in \Omega \tag{3.20}$$

$$q_{ij}^t,\ H_{ij}^t,\ w_{ij}^t \geq 0 \qquad \forall (i,j) \in E, t \in \{T_0, \ldots, T\} \tag{3.21}$$

$$q_{00}^t \leq 0 \qquad \forall t \in \{T_0, \ldots, T\} \tag{3.22}$$

$$X_{ij}^t,\ Y_{ij}^t \in \{0,1\} \qquad \forall (i,j) \in A^+, t \in \{T_0, \ldots, T\} \tag{3.23}$$

$$Q^t \geq 0 \qquad \forall t \in \{T_0, \ldots, T\} \tag{3.24}$$

$$S_{uv}^t \in \{0,1\} \qquad \forall (u,v) \in \Omega, t \in \{T_0, \ldots, T\} \tag{3.25}$$

The objective function (3.3) minimizes the total cost, which includes the routing cost and the penalty cost for not having the desired humidity level. It combines Eqs. (3.1) and (3.2). Constraints (3.4) through (3.8) are inventory-related constraints. The set of constraints (3.4) ensures that the value of w_{ij}^t is at least the amount of water needed to reach the desired level. Equation (3.5) sets the humidity level of edge (i, j) for the next time period, subtracting the water lost by traffic and evaporation $(e_t + f_{ij})$ and adding the water q_{ij}^t received when a service takes place. Constraints (3.6) set the initial humidity level at time 0. Constraint (3.7) ensures that the quantity delivered does not exceed the maximum humidity level. Constraints (3.8) ensure that the maximum humidity level, H_{ij}^{max}, is never exceeded.
Constraints (3.9) through (3.14) are capacity constraints. The set of constraint (3.9) ensures that the quantity delivered does not exceed the vehicle capacity in any time period. Equation (3.10) sets the initial vehicle level at time 0. Constraints (3.11) ensure that the vehicle capacity is not exceeded. Equations (3.12) indicate the water level of the vehicle in each time period. The water used for servicing is subtracted from the water level of the previous time period and the quantity replenished at the depot (q_{00}^t) is added when the vehicle returns for a refill. Constraints (3.13) and (3.14) are refilling constraints. Two refilling scenarios are considered : either the vehicle is completely empty or the remaining water is equal to the difference $Q^{max} - Q^t$. The variable q_{00}^t is set to the larger value. Note that q_{00}^t is always negative and will be added to the quantity of water in the vehicle in (3.12).
Constraints (3.15) through (3.20) are flow-conservation constraints. Constraints (3.15) ensure that no arc is both watered and traversed without watering. Equation (3.16) ensures that the vehicle starts at the depot, while equation (3.17) ensures that it ends at the depot in period T. Constraints (3.18) ensure the flow conservation. Whenever the vehicle traverses an edge from i to j (or from j to i) it must continue its route along an adjacent edge. The number of time periods is equal to the number of distance units because we assume that the speed is constant ; they are represented by d_{ij}. If the vehicle traverses edge (i, j) at the beginning of period t with or without service, it will choose an adjacent edge (j, k) to continue its travel after d_{ij} time periods. Constraints (3.19) ensure that while the vehicle is traversing an edge with or without service it remains in that edge for exactly d_{ij} time units. This is evident for one-unit-long edges, but presents a difficulty for edges that are traversed in more than one period of time. These edges will be separated from the rest. Set $\Omega \subset A$ contains all the arcs $(i, j) \in A$ whose length is greater than 1, i.e., $d_{ij} > 1$. The binary variable $S_{uv}^t = 1$ if the vehicle is traversing the arc $(u, v) \in \Omega$ at the beginning of time t, and 0 otherwise. Given m time units where $m \in \{1, \ldots, d_{ij} - 1\}$, constraints (3.20) ensure that the vehicle is not traversing or watering another edge.
The variables are defined in constraints (3.21) through (3.25).

3.5.4 Fixed-rate spraying

In some situations, the water vehicle is not able to deliver the exact quantity of water needed for an optimal solution. Therefore, we need to modify the original model. Variable q_{ij}^t, which represents the amount of water used for edge (i, j) at the beginning of time t, is replaced by a constant. We use a parameter $\alpha D d_{ij}$, where α is the number of liters of water per meter, D is the length of one distance unit, and d_{ij} is the number of distance units in arc (i, j). For example, if $\alpha = 3.6\,\mathrm{L/m}$ and each distance unit is $D = 500\,\mathrm{m}$, a single-unit edge $(d_{ij} = 1)$ would require 1800 L while a three-unit edge $(d_{ij} = 3)$ would require 5400 L. The constraints from the previous model involving variable q_{ij}^t are modified as follows :

– Constraints (5.31) are changed to

$$H_{ij}^t = (1 - (e_t + f_{ij}))H_{ij}^{t-1} + \alpha D(d_{ij}X_{ij}^t + d_{ji}X_{ji}^t) \qquad \forall (i, j) \in E,\ t \in \{T_0, \ldots, T\}. \quad (3.26)$$

– Constraints (3.7) are changed to

$$\alpha D(d_{ij}X_{ij}^t + d_{ji}X_{ji}^t) \leq H_{ij}^{max} \qquad \forall (i, j) \in E,\ t \in \{T_0, \ldots, T\}. \quad (3.27)$$

– Constraints (3.9) are changed to

$$\alpha D \sum_{(i,j)\in A^+} (d_{ij}X_{ij}^t) \leq Q^t \qquad \forall\, t \in \{T_0, \ldots, T\}. \quad (3.28)$$

– Constraints (3.12) are changed to the following equation. Note however that variable q_{00}^t is not removed because it does not indicate the quantity of water sprayed but instead the quantity of water used to refill the vehicle.

$$Q^{t+1} = Q^t - \alpha D \sum_{(i,j)\in A^+} (d_{ij}X_{ij}^t) - q_{00}^t \qquad t \in \{T_0, \ldots, T-1\}. \quad (3.29)$$

– Finally, variable q_{ij}^t is removed from nonnegativity set of constraints (3.21).

3.5.5 Different vehicle speeds

Suppose that the water vehicle has one speed for traversing an edge and another for watering, and assume that no other elements affect the truck speed. The watering speed is lower than the traversing speed because of the water delivery : the truck will take more time to traverse arc (i, j) when servicing it. If d_{ij} is the time (in number of periods) to traverse arc (i, j) without watering, the number of periods needed to water this arc will be λd_{ij}, where $\lambda \in \{1, 2, \ldots,\ \Lambda\}$. For example, if the watering speed is 20 km/h and the traversing speed is 40 km/h, $\lambda = 2$.

Consider Fig. 3.4. Edge (A, B) has a length $d_{AB} = 3$, and $\lambda = 2$. If the vehicle starts traversing edge (A, B) at time $t = 2$ and there is no service, it will begin traversing edge

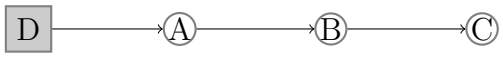

Figure 3.4 Example of different truck speeds

(B, C) at time $t = 5$, while if there is service, it will start edge (B, C) at time $t = 8$. Either edge (A, B) is watered or it is not. If it is not watered, then $Y^2_{AB} = 1$ and either $Y^5_{BC} = 1$ or $X^5_{BC} = 1$. If it is watered, then $X^2_{AB} = 1$ and either $Y^8_{BC} = 1$ or $X^8_{BC} = 1$. The following modification to (3.18) applies :

$$X^t_{ij} \leq \sum_{k|(j,k)\in A^+} X^{t+\lambda d_{ij}}_{jk} + Y^{t+\lambda d_{ij}}_{jk} \qquad \forall (i,j) \in A^+,\ t \in \{T_0, \ldots, T\}$$

$$Y^t_{ij} \leq \sum_{k|(j,k)\in A^+} X^{t+d_{ij}}_{jk} + Y^{t+d_{ij}}_{jk} \qquad \forall (i,j) \in A^+,\ t \in \{T_0, \ldots, T\}.$$

To ensure that the vehicle is not on a different edge when traversing an arc of length greater than 1, (3.19) can be separated into :

$$Y^t_{ij} \leq S^{t+m}_{ij} \qquad \forall (i,j) \in \Omega,\ m \in \{0, \ldots,\ d-1\},\ t \in \{T_0, \ldots,\ T-m\}$$

$$X^t_{ij} \leq S^{t+m}_{ij} \qquad \forall (i,j) \in \Omega,\ m \in \{0, \ldots,\ \lambda d-1\},\ t \in \{T_0, \ldots,\ T-m\}.$$

3.6 Test results

Since this is a new problem no benchmarks are available. We created a library of 10 instances specifically for the watering problem in open-pit mines. They are labeled *jpr* corresponding to the initials of the first author of this paper. Their shape resembles a tree, which is suitable for a mine network. We also used the 23 instances for CARP created by (Golden *et al.*, 1983) that are available at http ://www.uv.es/belengue/carp.html. This library was labeled *gdb*, corresponding to the initials of the authors. The instances were modified according to the characteristics of our problem, and the modified library is labeled *gdbj*. The model was coded in Cplex 12.4 and executed on a 2.27 GHz Intel Core i3 Notebook PC.
The procedure to create the *jpr* and *gdbj* instances is as follows :

– The cost of the *gdb* instances becomes the traversing cost for *gdbj*. Two units are added to the watering cost.

- The demand of the *gdb* instances is used to classify the roads according to their priority, where a higher demand means a higher penalty cost. The penalty cost P_{ij} has to be greater than or equal to the maximum watering cost, $\min P_{ij} \geq \max c_{ij}$, to ensure that the high-priority roads are serviced first. For the *jpr* instances, the priority is assigned to main roads rather than randomly. Figure 3.1a) shows an example of a main road in the network, road D-4.
- The water level and traversing cost are linked to the length of each edge. We assume that the longer the edge, the higher the traversing cost and the greater the need for water, since the number of liters needed to suppress dust is a function of the length of the road. Using 3.6 L/m (voir Li *et al.*, 2008, page 9) as the watering rate and 300 m as the unit of distance, we calculated the required water level, $\overline{h_{ij}}$, as 1080 L per distance unit. We then reduced this to 60% of the original value in order to allow a service frequency of thirty minutes for high-priority roads and one hour for low-priority roads.
- The initial level, L_{ij}, is set at 10% of the required level, and the maximum level, H_{ij}^{max}, is set to 10% above the required level. The evaporation percentage, $(f_{ij} + e_t)$, is set to 3% to 10% depending on the importance of the road.
- Instances *jpr1* through *jpr5* are trees, while *jpr6* through *jpr10* resemble trees with a cycle at the end of one or several branches.

3.6.1 Limits on the number of periods

The number of variables, and thus the execution time, are affected by both the number of arcs and the number of time periods. Each instance can be run for any number of periods. A small number may cause the vehicle to avoid important arcs if they are located far from the depot or are too long. On the other hand, a large number will result in long computational times. Based on the distance of the arcs from the depot, we established the number of time periods for each network. The longer the edges, the greater the number of time periods they require. To illustrate this situation, consider the network in Fig. 3.5 corresponding to instance *gdbj19*. For edge $(3, 7)$, whose length is 6 units, the total distance needed to arrive, traverse the edge, and return to the depot is the shortest path to either node 3 or 7 plus the length of the edge. The total distance is therefore 17 units. The parameter μ is the minimum number of time periods the vehicle needs to traverse the farthest edges of the network and return to the depot. If edge $(3, 7)$ is the farthest edge in the network, then $\mu = 17$. This means that in 17 time units, the vehicle should be able to traverse any edge and return to the depot. It will not necessarily traverse edge $(3, 7)$—that depends on its priority—but with fewer time periods this edge cannot be serviced. The process described for edge $(3, 7)$ is done for each edge in the network, and the maximum value is taken as μ. We used Dijkstra's algorithm to

calculate the shortest path to each node. Note that the number of periods is determined by the length of the edges and not by the number of edges in the network.

We run the program for a number of periods $p \geq \mu$. In theory, p could be increased indefi-

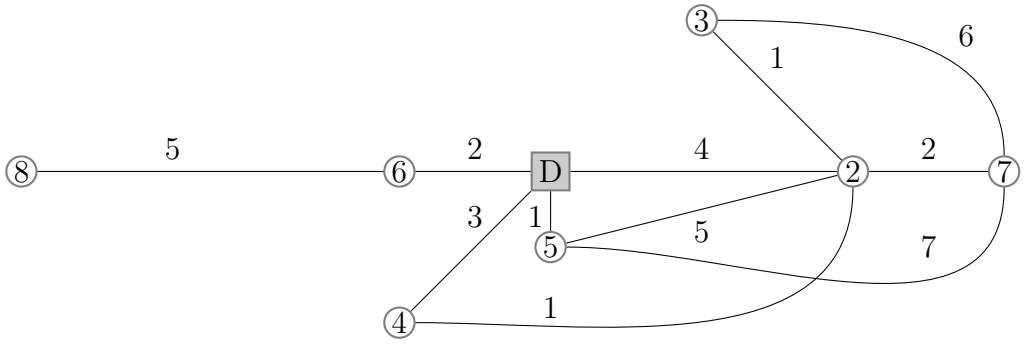

Figure 3.5 Representation of network *gdbj19*.

nitely, thus increasing the computational time ; we established an upper limit. Suppose each time unit is one minute in real time. The program will not be practical if it takes more time to calculate an optimal route that the actual time it takes for the vehicle to perform that route. The time limit (in minutes) is therefore the same as the number of time periods. For example, the time limit for instance *gdbj19* is 1020 s. If the program requires longer then we consider that it cannot solve the instance. Tables 3.1 and 3.2 show the values for μ and the time limits.

3.6.2 Comparison of models with fixed-rate and variable-rate spraying

We compared the results of two versions of the model. The first version, hereafter called model A, has variable-rate spraying, constraints (3.3) through (3.25). The second version, hereafter called model B, has fixed-rate spraying using the modifications (3.26) through (3.29).

To reduce the computational time, we introduced a parameter sp_{ij} that indicates the number of time periods needed to arrive at edge (i, j), i.e., the shortest path from the depot to node i or j. During this period, the vehicle has not arrived at edge (i, j) and therefore, cannot traverse it. We set $X_{ij}^t = 0$ and $Y_{ij}^t = 0$ for all $(i, j) \in A$, $t \in \{T_0, \ldots, sp_{ij}\}$.

Tables 3.3 and 3.4 compare models A and B when they were run for μ time periods. For each model, the columns indicate the number of variables (Var), the solution found, and the computational time. The optimal solution is expressed in tens of thousands of units. The

Tableau 3.1 Values of μ and time limit for *gdbj* instances.

Network gdbj	Nodes	Edges	μ	Time limit (s)
1	12	22	63	3780
2	12	26	59	3540
3	12	22	59	3540
4	11	19	64	3840
5	13	26	64	3840
6	12	22	64	3840
7	12	22	58	3480
8	27	46	38	2280
9	27	51	37	2220
10	12	25	39	2340
11	22	45	43	2580
12	13	23	89	5340
13	10	28	128	7680
14	7	21	15	900
15	7	21	8	480
16	8	28	14	840
17	8	28	9	540
18	9	36	10	600
19	8	11	17	1020
20	11	22	20	1200
21	11	33	15	900
22	11	44	12	720
23	11	55	13	780

solutions marked by (*) are integer feasible solutions found within the time limit, but there is no indication that they are optimal. The other solutions are optimal. No feasible solution was found for network *gdbj13*.

For every network, the computational time is shorter with model B, which can be explained by the reduction in the number of variables. This reduction also explains why the maximum number of periods for which an optimal solution can be found is higher for model B. The optimal objective, however, is always lower for model A. This is because the vehicle is allowed to use any quantity of water up to the maximum level, thus allowing the humidity level to decrease more slowly. On the other hand, the fixed quantity cannot be added more than once in order to prevent over-watering.

Tableau 3.2 Values of μ and time limit for *jpr* instances.

Network gdbj	Nodes	Edges	μ	Time limit (s)
1	7	6	30	1800
2	11	10	32	1920
3	13	12	33	1980
4	12	11	19	1140
5	17	16	20	1200
6	11	11	37	2220
7	17	21	17	1020
8	19	22	40	2400
9	22	22	37	2220
10	21	23	32	1920

We performed a graph transformation on networks *gdbj1* through *gdbj7* and *gdbj10* through *gdbj12* to reduce the computational time and find an optimal solution. In these networks, the arc length is greater than 1, i.e., set $\Omega = A$. The arc length is reduced by a factor equal to the minimum arc distance found in the network, 3 for *gdbj12* and 2 for the others. For example, an edge that originally had 20 distance units of 300 m was transformed into an edge of 10 distance units of 600 m. In real time, it will take 2 min to traverse a single distance unit instead of the 1 min it took before the transformation. All the other parameters are unchanged, and decimals are rounded down. The results for the transformed network are shown in Table 3.5. The value of μ has been reduced since fewer time periods needed to reach the farthest edges of the network. The optimal solution was found using either Model A or Model B for all the instances except *gdbj6*, and the computational times were significantly reduced.

We increased the number of periods and the time limit to find the maximum value for which the models could compute an optimal solution within the limits. The results for the *gdbj* and *jpr* instances are shown in Tables 3.6 and 3.7 respectively.

For every network, the maximum number of time periods for which a solution can be found is greater for model B than for model A. This is because the computational time is lower for model B than for model A when the number of periods is the same. The only exception is *jpr4*, for which the maximum number of periods is 33 for both models.

Networks *jpr1* and *jpr2* do not have a maximum number of periods. Their small size and edge length allow a single vehicle to service the complete set of arcs in less time than that required for the water to evaporate.

Tableau 3.3 Computational results for *gdbj* instances.

Network gdbj	μ	Model A			Model B		
		Var	Solution (x10000)	Time (s)	Var	Solution (x10000)	Time (s)
1	63	16969	185750.23*	3780	15121	187884.95*	3780
2	59	19041	231885.81*	3540	16961	232863.04*	3540
3	59	16161	180294.70*	3540	14401	180341.61*	3540
4	64	14876	201153.18	3069.41	13261	203025.70	2030.87
5	64	20707	276183.58*	3840	18445	278405.45*	3840
6	64	17575	222559.59*	3840	15661	224677.63*	3840
7	58	15959	201921.29	2535.42	14221	203158.48	2154.30
8	38	19177	215461.88*	2280	17015	218877.487*	2280
9	37	18943	184483.49*	2220	16801	184697.911*	2220
10	39	13512	149966.21	88.75	12037	150078.73	79.54
11	43	25768	851923.82*	2580	22933	854788.40*	2580
12	89	25321	4971783.00*	5340	22561	5001627.51*	5340
13	62	46981	—	3720	41797	—	3720
14	15	4489	3253.05	10.36	3985	3254.70	7.60
15	8	2716	1026.86	1.55	2401	1026.86	0.13
16	14	5751	4109.13	12.01	5107	4110.88	10.97
17	9	4115	2281.03	3.90	3639	2281.03	2.50
18	10	5967	3898.17	0.52	5283	3898.17	0.52
19	17	2674	2142.97	2.21	2377	2148.59	2.20
20	20	5941	5690.60	155.64	5281	5709.07	144.30
21	15	7276	5762.31	27.64	6451	5766.55	12.71
22	12	8229	6995.20	34.24	7261	6996.36	12.81
23	13	10539	7319.54	198.01	9329	7320.95	146.47

The difference between models A and B becomes evident when we analyze the resulting routes. Table 3.8 shows the edges visited by the vehicle, their lengths, their priorities, and the quantities of water delivered if service took place. The greater the priority, the higher the importance of the edge. The vehicle follows exactly the same route in both examples, but the quantity delivered is different. In Table 3.8a) the quantity is slightly greater than in 3.8b), which allows the humidity to remain above the required level for longer and thus reduces the objective value.

For model B, the restrictions on the quantity delivered make it difficult to always follow the same route. In Table 3.9a) edge $(0, 5)$ is visited twice : in period 1 and then in period 25. This does not mean that the service frequency is 25 periods, but that the quantity delivered

Tableau 3.4 Computational results for *jpr* instances.

Network gdbj	μ	Model A			Model B		
		Var	Solution (x10000)	Time (s)	Var	Solution (x10000)	Time (s)
1	30	2301	1055.81	2.64	2025	1394.61	2.25
2	32	3527	424.96	2.89	3097	715.19	2.84
3	33	4401	1020.99	10.45	3873	1269.38	8.55
4	19	2140	196.25	4.15	1887	210.06	3.40
5	20	3217	250.81	12.73	2833	269.02	10.16
6	38	4654	1393.71	27.13	4137	1419.92	23.56
7	17	3550	644.55	47.95	3109	653.40	34.71
8	40	8209	1856.38	258.96	7201	2260.44	171.88
9	37	8367	3917.61	269.15	7333	4351.11	238.23
10	32	6765	1921.29	506.93	5929	1982.65	201.01

Tableau 3.5 Computational results for *gdbj* instances after distance transformation.

Network gdbj	μ	Model A			Model B		
		Var	Solution (x10000)	Time (s)	Var	Solution (x10000)	Time (s)
1	30	8037	8182.82	1274.98	7135	8638.67	643.39
2	28	9049	10251.90	2811.78	8035	10739.50	1065.47
3	28	7645	7911.56	1520.56	6787	8399.13	558.69
4	31	7183	8998.26	525.63	6385	9505.45	211.32
5	30	9745	12074.40	2322.34	8653	12592.40	845.90
6	30	8233	—	—	7309	10163.63	774.15
7	27	7449	8796.64	85.65	6613	9237.18	22.89
10	18	6301	6646.87	7.53	5601	6870.58	4.24
11	20	11431	37115.30	223.81	10081	38298.03	31.08
12	26	7660	122266.96	3498.59	6809	135182.10	179.64

can be split into two deliveries. This is an advantage of model A that model B does not allow because of the maximum level of humidity that the edge can handle, so a different route is shown in 3.9b).

An example of the service frequency is shown in Table 3.9. There is a second service for edge $(0, 2)$ after 36 and 34 time periods in 9a) and 9b) respectively. There is an increase in the deadheading in model B for the edges that were previously serviced $((11, 19), (10, 11), (2, 10))$.

Tableau 3.6 Maximum number of periods for which models A and B find an optimal solution within the time limit for *gdbj* instances.

Network	Model A			Model B		
gdbj	max p	Solution (x10000)	Time (s)	max p	Solution (x10000)	Time (s)
1	30	8182.82	1274.98	32	9201.37	1184.27
2	28	10251.90	2811.78	31	11882.61	3380.34
3	28	7911.56	1520.56	31	9281.02	2167.18
4	34	9883.97	2854.99	38	11625.86	4012.19
5	30	12074.40	2322.34	32	13425.95	2734.84
6	—	—	—	32	10829.12	1603.00
7	32	10400.36	1601.37	34	12258.70	4031.98
10	26	9119.82	1102.58	30	11081.70	2927.28
11	24	44406.59	1370.31	27	51948.82	2471.90
12	26	122266.96	3498.59	33	166782.30	5751.69
14	23	4766.01	659.65	26	5427.41	1334.20
15	18	2120.80	366.99	20	2333.22	1098.21
16	21	5981.42	824.42	22	6188.72	863.58
17	18	4406.57	395.42	19	4651.23	644.99
18	25	8978.59	919.17	27	9573.53	1564.22
19	30	3515.41	1759.83	31	3731.04	828.23
20	24	6803.39	925.98	25	7086.40	1177.39
21	20	7742.18	644.02	22	8821.59	856.81
22	17	9896.54	444.62	20	11558.84	790.55
23	13	7319.54	198.01	16	9045.59	352.50

In model A, these edges receive a small quantity the second time that they are serviced, but in model B spraying the same quantity again would produce over-watering. The use of a variable quantity leads to an advantage in the number of edges serviced. The route of model A (3.9a) includes more edges than that of model B (3.9b). Edge $(11, 20)$ is not included in the latter route since the cost of not watering it is less than the cost of not watering edge $(0, 2)$ a second time.

Notice the effect of the priority parameter. The highest priority given in network *jpr8* is 5, so the vehicle follows the route with these high-priority edges. In the *gdbj* instances the priority was assigned randomly instead of to the main roads.

The effect of a reduction of the vehicle capacity is shown in Table 3.11. Network *gdb6* is solved, in both cases, with model B for 37 time periods. The vehicle follows the high-priority edges as expected. The only difference between the two results is the vehicle capacity. The

Tableau 3.7 Maximum number of periods for which models A and B find an optimal solution within the time limit for *jpr* instances.

Network	Model A			Model B		
jpr	max p	Solution (x10000)	Time (s)	max p	Solution (x10000)	Time (s)
1	—	—	—	—	—	—
2	—	—	—	—	—	—
3	43	1298.18	560.47	45	1739.03	1882.17
4	33	265.56	1404.57	33	329.99	1203.30
5	30	322.66	1203.07	31	378.78	1330.75
6	46	1581.52	1681.14	49	1780.20	2701.83
7	25	873.60	1132.97	26	942.78	720.29
8	48	2080.05	577.17	50	2651.22	2815.41
9	40	4259.66	900.39	43	4896.64	1852.34
10	35	1968.40	665.27	37	2253.87	2210.01

Tableau 3.8 Comparison of routes for *gdbj20* network from models A and B.

a. *gdbj20* final route using model A					b. *gdbj20* final route using model B				
Period	Edge	Length	Priority	Quantity	Period	Edge	Length	Priority	Quantity
1	< 0 7 >	1	7	1087.56	1	< 0 7 >	1	7	1080
2	< 7 8 >	3	5	3283.77	2	< 7 8 >	3	5	3240
5	< 8 0 >	9	8	10015.79	5	< 8 0 >	9	8	9720
14	< 0 9 >	1	1	808.23	14	< 0 9 >	1	1	1080
15	< 9 4 >	6	9	6909.82	15	< 9 4 >	6	9	6480
21	< 4 0 >	3	1	3393.09	21	< 4 0 >	3	1	3240
24	< 0 0 >	1	—	0	24	< 0 0 >	1	—	0

capacity for the route shown in Table 3.11b) is a quarter of that in 3.11a). The capacity was reduced to show the use of arc $(0, 0)$, which serves as a refill point for the vehicle at the depot before it starts a new route in the same time horizon. The vehicle visits the depot twice during the route, in periods 7 and 22. The quantity (the value of variable q^t_{00}) is negative. Once the vehicle is replenished, it continues its deliveries.

Tableau 3.9 Comparison of routes for *gdbj19* network from models A and B.

a. gdbj19 final route using model A					b. gdbj19 final route using model B				
Period	Edge	Length	Priority	Quantity	Period	Edge	Length	Priority	Quantity
1	< 0 5 >	1	5	1087.56	1	< 0 5 >	1	5	1080
2	< 5 7 >	7	9	7662.13	2	< 5 7 >	7	9	7560
9	< 7 2 >	2	9	2263.59	9	< 7 3 >	6	8	6480
11	< 2 3 >	1	4	1078.63	15	< 3 2 >	1	4	1080
12	< 3 7 >	6	8	6856.75	16	< 2 4 >	9	6	9720
18	< 7 5 >	7	9	5711.99	25	< 4 0 >	3	3	3240
25	< 5 0 >	1	5	723.28	28	< 0 0 >	1	—	0
26	< 0 6 >	2	8	1699.76	29	< 0 0 >	1	—	0
28	< 6 0 >	2	8	0	30	< 0 0 >	1	—	0
30	< 0 0 >	1	—	0					

Tableau 3.10 Comparison of routes for *jpr8* network from models A and B.

a. jpr8 final route using model A					b. jpr8 final route using model B				
Period	Edge	Length	Priority	Quantity	Period	Edge	Length	Priority	Quantity
1	< 0 2 >	7	5	7612.92	1	< 0 2 >	7	5	7560
8	< 2 10 >	5	5	5637.82	8	< 2 10 >	5	5	5400
13	< 10 11 >	7	5	8021.69	13	< 10 11 >	7	5	7560
20	< 11 19 >	1	3	1129.27	20	< 11 19 >	1	3	1080
21	< 19 11 >	1	3	35.64	21	< 19 11 >	1	3	0
22	< 11 20 >	1	1	1132.74	22	< 11 10 >	7	5	0
23	< 20 11 >	1	1	35.64	29	< 10 2 >	5	5	0
24	< 11 10 >	7	5	4572.93	34	< 2 0 >	7	5	7560
31	< 10 2 >	5	5	4820.83	41	< 0 0 >	1	—	0
36	< 2 0 >	7	5	7660.13	42	< 0 0 >	1	—	0
43	< 0 0 >	1	—	0	43	< 0 0 >	1	—	0
44	< 0 0 >	1	—	0	44	< 0 0 >	1	—	0
45	< 0 0 >	1	—	0	45	< 0 0 >	1	—	0

3.6.3 Computational time

We studied the computational times of model A and B for the 20 *gdbj* instances and the 10 *jpr* instances for which an optimal solution was found. We measured the time taken to achieve a relative gap of 2%. The relative gap measures the difference between the integer solution and the lower bound of the linear relaxation. The program stops when this parameter reaches 0.01%, but it could be set to higher values. However, when we increase the gap we

Tableau 3.11 Comparison of different vehicle capacities for *jpr6* instance using model B.

a. jpr6 final route using model B					b. jpr6 final route using model B				
Period	Edge	Length	Priority	Quantity	Period	Edge	Length	Priority	Quantity
1	< 0 6 >	2	8	2160	1	< 0 4 >	3	8	3240
3	< 6 0 >	2	8	0	4	< 4 0 >	3	8	0
5	< 0 4 >	3	8	3240	7	< 0 0 >	1	—	-2960
8	< 4 0 >	3	8	0	8	< 0 2 >	4	8	4320
11	< 0 5 >	1	8	1080	12	< 2 0 >	4	8	0
12	< 5 0 >	1	8	0	16	< 0 5 >	1	8	1080
13	< 0 2 >	4	8	4320	17	< 5 0 >	1	8	0
17	< 2 3 >	1	1	1080	18	< 0 6 >	2	8	2160
18	< 3 2 >	1	1	0	20	< 6 0 >	2	8	0
19	< 2 0 >	4	8	0	22	< 0 0 >	1	—	-5400
23	< 0 6 >	2	8	0	23	< 0 6 >	2	8	0
25	< 6 8 >	5	1	5400	25	< 6 9 >	2	1	2160
30	< 8 6 >	5	1	0	27	< 9 6 >	2	1	0
35	< 6 0 >	2	8	2160	29	< 6 0 >	2	8	0
37	< 0 0 >	1	—	0	31	< 0 4 >	3	8	0
					34	< 4 0 >	3	8	3240
					37	< 0 0 >	1	—	0

may not find the optimal solution. For all the instances tested, the optimal solution was found before the gap reached 2%. Tables 3.12 and 3.13 show the percentage reduction in the computational time when the program is stopped at a 2% gap.

For the *gdbj* instances the average time reduction is 2.02% for model A and 3.22% for model B. For the *jpr* instances the average reductions are 5.08% and 3.35% respectively. The optimal solution was known and there was no loss of optimality when the program was stopped at the 2% relative gap. The type of model (A or B) has little influence, but the *jpr* instances in general have greater reductions. The difference is greater than 2% for 7 (6) of the *jpr* instances for model A (B). In some cases such as *jpr7* the difference is above 15%. The difference is greater than 2% for 9 (15) of the *gdbj* instances for model A (B).

Tableau 3.12 Percentage reduction in computational time when stopping criterion set to 2% relative gap for *gdbj* instances.

Network gdbj	Number of periods	Model A		Model B	
		Total time	% reduction	Total time	% reduction
1	30	1274.98	0.00%	643.39	7.18%
2	28	2811.78	2.35%	1065.47	3.95%
3	28	1520.56	1.03%	558.69	1.87%
4	31	525.63	1.02%	211.32	3.29%
5	30	2322.34	2.02%	845.90	3.34%
6	30	3758.56	0.95%	774.15	2.25%
7	27	85.65	3.74%	22.89	4.15%
10	26	1102.58	1.64%	325.07	3.77%
11	24	1370.31	1.75%	347.79	9.22%
12	26	3498.59	1.98%	179.64	6.50%
14	23	694.27	2.31%	384.98	0.27%
15	18	366.99	1.76%	308.23	2.42%
16	21	824.42	8.07%	509.41	0.95%
17	18	395.42	2.48%	376.85	1.24%
18	25	919.17	0.44%	831.78	2.42%
19	30	1759.83	1.12%	662.51	2.25%
20	24	925.98	2.26%	772.33	2.52%
21	20	644.02	1.13%	326.35	2.57%
22	17	444.62	2.43%	266.42	0.97%
23	13	198.01	2.03%	146.47	3.29%
Average			2.02%		3.22%

3.7 Conclusion and future work

Two mathematical models were developed for the PCARP with inventory constraints. These models can be applied to the road watering problem in open-pit mines. The first model solves a routing problem combined with an inventory problem. It determines the edges to be serviced and the quantity to be delivered to each. The second model, a more restricted version, determines the route to follow when the rate at which the water is sprayed is fixed. The models respond correctly to changes in the parameters such as different vehicle capacities, edge priorities, and evaporation rates. The first model has a smaller optimal objective because it is less restricted than the second model, but its larger number of variables results in a longer computational time.

Tableau 3.13 Percentage reduction in computational time when stopping criterion set to 2% relative gap for *jpr* instances.

Network gdbj	Number of periods	Model A		Model B	
		Total time	% reduction	Total time	% reduction
1	45	84.35	3.86%	60.14	2.44%
2	47	167.01	11.63%	146.50	4.55%
3	40	355.26	7.44%	302.32	1.26%
4	25	131.49	0.37%	62.42	0.16%
5	27	414.67	0.55%	318.98	2.36%
6	43	220.51	2.25%	74.12	3.17%
7	25	1132.97	7.42%	529.26	15.88%
8	45	577.17	8.09%	504.04	1.17%
9	40	900.39	7.32%	611.76	0.43%
10	35	1179.65	1.84%	742.38	2.08%
average			5.08%		3.35%

The models we tested (A and B) can solve instances with up to 22 nodes and 40 to 55 edges when the number of time periods is 20 to 30. In some cases, problems with 40 to 45 time periods can be solved. The computational time is affected not only by the size of the network, but also by the geometry of the graph. The length of the edges and their distance from the depot influence the number of periods needed for the vehicle to traverse the network. A larger number of time periods increases the number of variables and the computational time. The relatively small *gdbj13* network (28 edges) was not solved because the vehicle required a large number of time periods to reach all the edges. On the other hand, optimal solutions were found for larger networks such as *gdbj23* (55 edges) because all the edges could be reached from the depot in a shorter time. Stopping the program at the 2% gap resulted in a 2% to 5% reduction in the computational time without loss of optimality ; this approach could be used to evaluate larger networks.

The time horizon of the problem under study is continuous, hence, the importance of obtaining a good solution for a large number of time periods. However, the size of the problem is increased with the number of periods. Therefore, other solution methods need to be explored, i.e. a heuristic or metaheuristic approach.

Future work will also focus on finding a solution when the problem has more than one vehicle and more than one depot. The inclusion of more than one vehicle is not trivial because it increases the complexity of the algorithm. A more complex model is considered to be addres-

sed using a different approach, and thus the use of heuristic or metaheuristic methods may be considered.
To the best of our knowledge, this is the first mathematical model for an arc routing problem that includes routing and inventory decisions.

Acknowledgements

This project was partly funded by the Mexican Council of Science and Technology (CONACYT) and the Group for Research in Decision Analysis (GERAD).
We are grateful to the reviewer who contributed to the improvement of this paper.

CHAPITRE 4

ARTICLE 2 : ADAPTIVE LARGE NEIGHBORHOOD SEARCH FOR THE PERIODIC CAPACITATED ARC ROUTING PROBLEM WITH INVENTORY CONSTRAINTS

Cet article, (Riquelme Rodríguez *et al.*, 2014) a été soumis pour revue dans le journal *Networks*, le 12 décembre 2013. On a reçu les commentaires des arbitres le 17 février 2014 et on a répondu le 28 février 2014.

Cet article étend le problème présenté dans la section précédente de un à plusieurs véhicules. Dans le contexte de problèmes périodiques de tournées sur les arcs, l'application dans laquelle on travaille est l'arrosage des chemins de terre dans les mines à ciel ouvert. Le modèle mathématique considère une flotte de K camions-citerne qui servent les arêtes du réseau. L'inclusion de plus d'un véhicule augmente le nombre de variables dans le modèle mathématique par rapport à celui du modèle présenté pour un véhicule d'arrosage. Par conséquent, le modèle est capable de résoudre seulement de très petits réseaux pour un horizon de temps réduit. Un algorithme appelé *adaptive large neighborhood search* (ALNS) est utilisé pour résoudre les réseaux de grande taille pour des horizons de temps plus longs. Premièrement, un algorithme de construction est utilisé pour créer une solution initiale, laquelle est modifiée par un ensemble d'opérateurs de type destruction-réparation qui sont combinés de telle façon que celui qui a la meilleure performance a une plus forte probabilité d'être choisi pour une nouvelle itération. L'algorithme est testé pour un ensemble d'exemplaires de CARP connus. Il est également testé pour un ensemble d'exemplaires développés à partir des mines à ciel ouvert réelles. On montre la performance des opérateurs destruction-réparation utilisés individuellement, ainsi que la performance de l'ALNS.

Comme il sera expliqué dans l'article, l'utilisation de contraintes de gestion de stocks affecte l'utilisation des opérateurs de destruction-réparation l'ALNS. Était donné que ces contraintes portent sur l'inventaire et les coûts d'une période de temps à l'autre, au moment où une solution est détruite, elle ne peut pas être réparée à nouveau à moins que tous les trajets suivants ainsi que les coûts soient recalculés.

Les contributions de cet article se résument comme suit :

– On présente un modèle mathématique qui tient compte de l'utilisation de plus d'un

véhicule pour le problème d'arrosage dans les mines à ciel ouvert.

– L'application de l'algorithme ALNS avec huit opérateurs de destruction-réparation au problème avec des contraintes de gestion de stocks présente un défi unique pour le processus de modification de la solution existante.
– Le réseau de cinq mines à ciel ouvert réels est utilisé pour tester l'algorithme ALNS et définir la taille des exemplaires qui peuvent être résolus dans un certain temps de calcul limite.

Adaptive large neighborhood search for the periodic capacitated arc routing problem with inventory constraints

Juan Pablo Riquelme Rodríguez, Michel Gamache, André Langevin
CIRRELT, GERAD and Department of Mathematics and Industrial Engineering
École Polytechnique de Montréal

4.1 Abstract

This article describes the problem in which the edges of a network represent customers, and a quantity of material is delivered to them so that each one achieves a desired inventory level while finding the lowest-cost route of delivery. Routing and inventory decisions are made at the same time. An example of an application of this problem is dust suppression in open-pit mines. A fleet of vehicles spray water along the roads of a mine. Humidity increases the effectiveness of dust-particle retention. Because the level of humidity decreases, replenishment is done periodically. Other examples of applications include dust suppression in forest roads and plants watering in street medians and sidewalks. We develop a mathematical model that combines two objectives : An inventory objective that minimizes the penalty for the lack of humidity and a routing objective that minimizes watering and traversing costs. Due to the complexity of the mathematical model, we developed an adaptive large neighborhood search algorithm that combines several destroy and repair operators dynamically.

Keywords : Periodic capacitated arc routing problem, vehicle routing, ALNS, open-pit mine.

4.2 Introduction

An arc routing problem is a routing problem in which the service activity takes place on the arcs of a network (Assad et Golden, 1995). A special case in the category of arc routing problems, is the periodic capacitated arc routing problem (PCARP). This problem was first described in (Lacomme *et al.*, 2002) for a garbage collection problem. The vehicles in charge of providing the service to the arcs, in this case, garbage trucks, have a limited capacity. Because garbage accumulates at different rates on different streets, the frequency of service varies. Thus, some of the streets visited on the first day, may not be visited on subsequent days. This means that the solution found for one period of time does not apply to the rest of them in a given time horizon. Therefore, a suitable solution for the whole time horizon is needed. A periodic problem is characterized by a different frequency of service depending on the customer's needs and the service spacing between visits.

PCARP was shown to be NP-hard (Lacomme *et al.*, 2002) because it includes the capacitated arc routing problem (CARP) as a particular case when the number of periods is one. CARP was shown to be NP-hard in (Golden et Wong, 1981).

Other applications of PCARP include road monitoring (Marzolf *et al.*, 2006) and road watering (Li *et al.*, 2008). Mathematical models for PCARP applications were proposed in (Chu *et al.*, 2005) and (Mei *et al.*, 2011) for the garbage collection problem and in Monroy *et al.* (2011) for the PCARP with irregular services. Because of the complexity of the problem, heuristic algorithms were also proposed to solve large instances. Heuristic methods for the PCARP include the algorithm for the periodic rural postman problem (Ghiani *et al.*, 2005), a memetic algorithm (Lacomme *et al.*, 2005), a scatter search algorithm (Chu *et al.*, 2006), and an ant colony heuristic (Kansou et Yassine, 2009).

Contrary to the garbage collection example, where the material is accumulated over time and then removed, we consider a problem in which material is delivered by a fleet of vehicles and then consumed over time, as in inventory management. This material is delivered to the edges of a network periodically. The delivery frequency depends on the edge's priority. Applications of this problem are road watering in open-pit mines, dust suppression in forest roads, and plant watering on street medians and sidewalks. We focus on the road watering problem in open-pit mines. This problem was first addressed by Li et al. (Li *et al.*, 2008) and will be described in the next section. Their objective is to minimize routing costs and delay costs while maintaining a fixed frequency of service on the edges. Our main objective is to reduce the costs associated with not-watered roads while reducing routing costs, considering the edges as customers with a level of inventory and focusing on the factors that make this inventory change through time.

This article is organized as follows : Section 4.3 defines the problem and presents the mathematical model. The heuristic method used to solve the problem is detailed in Section 4.4. Section 4.5 shows the results obtained by the algorithm on several instances. Finally, the conclusions are presented in Section 4.6.

4.3 Problem definition

4.3.1 The problem of watering roads in open-pit mines

When a hauling truck travels along the network of unpaved roads of an open-pit mine, dust particles are lifted from the road and become airborne. These particles damage vehicles and equipment, reduce visibility and, when inhaled, are harmful for workers (Li *et al.*, 2008). Due to the temporary nature of the roads, permanent dust suppression methods, such as

pavement, result in high costs (Neulicht et Shular, 1998). Thus, using trucks to spread water over the surface of the road, becomes the easiest dust-suppressing method due, in part, to the small amount of preparation roads need in order to be treated (Cecala *et al.*, 2012). Because water evaporates, watering must be done periodically. Evaporation can be increased by weather conditions and traffic volume. Those roads with higher traffic volume suffer a quicker evaporation. Therefore, they are classified according to the number of vehicles that travel along them. The roads with higher traffic volume are given a higher priority.

A water depot, located in the mine network, serves as the refill point from where trucks depart at the beginning of the time horizon and to where they return at the end. At any point within the time horizon, trucks can return to the depot to reload and start another route.

The humidity level in each road can be modeled using an economic order quantity model (EOQ) where demand is a function of time and the existing humidity level. Demand is partially due to evaporation which can be modeled using the results presented in Molina-Martínez *et al.* (2006) that simulate the hourly pan-evaporation of water during day-time. There is no holding cost, however, there is a maximum inventory level allowed. Shortage is represented by having a humidity level below the required level to ensure particle retention.

4.3.2 Mathematical model

The road network of an open-pit mine can be modeled as a mixed network $G = (N, E \cup A)$ where N is the set of nodes ; E is the set of edges that represent the network of roads of a mine ; and A is the set of arcs that represent the direction in which the edges are traversed. For each $[i, j] \in E$ there are two arcs $(i, j), (j, i) \in A$. Node 0 represents a depot. An artificial arc $(0, 0)$ forms a loop at the depot. Set A' is the set of arcs such that $A' = A - (0, 0)$.

Consider a time horizon representing a working shift, divided in time periods t of equal duration, $t \in \{0, \ldots, T\}$. Consider a fleet of K water trucks of capacity Q^{max}. For each edge $[i, j] \in E$ there is minimum level of humidity to ensure particle retention, $\overline{h_{ij}}$. A shortage cost P_{ij} is assigned to edge $[i, j]$ for having a level of humidity below $\overline{h_{ij}}$. The initial inventory H_{ij}^{int} is the level of humidity of edge $[i, j]$ at the beginning of the time horizon. H_{ij}^{max} is the maximum inventory level that corresponds to the maximum humidity level of edge $[i, j]$. Humidity is consumed by evaporation, which is caused by weather conditions and traffic volume. e_t represents the evaporation factor at the beginning of time t, while b_{ij} is the humidity consumption of edge $[i, j]$ due to traffic volume. Each edge is assigned a priority depending on the road's traffic volume. Frequently traversed roads have a higher priority, and thus the value of b_{ij} increases. r_{ij} is the cost of dead-heading of arc (i, j) and c_{ij} is the cost of watering

arc (i, j), which includes the cost of traversing. In the case of $(0, 0)$ both costs represent the waiting time and the cost of re-load. d_{ij} is the number of time periods required to traverse arc (i, j). Consider $\Omega \subseteq A$ the set of arcs that need more than one time period to be traversed, i.e., $d_{ij} > 1$. We consider parameter d_{ij} be the same weather the truck is servicing or not. Given ideal conditions, the truck's speed is different for service and deadheading. The road condition and the presence of hauling trucks or other vehicles on the same road significantly reduce the deadheading time of water trucks (Li *et al.*, 2008). The difference in both speeds was mentioned in Riquelme Rodríguez *et al.* (2013). The authors present an approach to model different truck speed when the deadheading travel time is less than the service time. This approach, however, is restricted to the deadheading speed being a multiple of the service speed.

Variable H_{ij}^t indicates the humidity level of edge $[i, j]$ at the beginning of time t. The shortage quantity for each edge $[i, j]$ at the beginning of time t is expressed by variable $w_{ij} = \max\{0, \overline{h_{ij}} - H_{ij}^t\}$. Variable Q^{kt} represents the level of water in truck k at time t. Because any truck can service any given edge, the quantity delivered to edge $[i, j]$ by truck k at a time t is denoted by variable q_{ij}^{kt}. $Y_{ij}^{kt} = 1$ if truck k starts traversing arc (i, j) at the beginning of time t without providing service (deadheading), and 0 otherwise. $X_{ij}^{kt} = 1$ if truck k starts watering arc (i, j) at the beginning of time t, and 0 otherwise. In order to reload the truck at the depot, variable q_{00}^{kt} denotes the quantity of water added to the truck at arc $(0, 0)$. Binary variable $S_{uv}^{kt} = 1$ if the truck k is traversing arc $(u, v) \in \Omega$, and 0 otherwise.

Consider Figure 3.2a where the inventory level of edge $[i, j]$, H_{ij}^t, varies over tieme and a quantity of water q_{ij}^{kt} is delivered at certain time t by truck k. It is gradually consumed over time due to evaporation and traffic volume. Figure 3.2b shows the same situation when the time horizon is divided in periods of equal length. We consider that water is delivered at the beginning of the time period. We also consider that the humidity level is constant through the duration of a time period. The number of liters per meter that are still needed to ensure particle retention is represented as the area under the line $\overline{h_{ij}}$. For edge $[i, j]$, the area for the complete time horizon is calculated using $\sum_{T_0}^{T} \max\{0, \overline{h_{ij}} - H_{ij}^t\}$. The objective is to minimize the total shortage cost, P_{ij} associated to this area.

$$\min \sum_{[i,j]\in E} \sum_{t=0}^{T} P_{ij} \max\{0, \overline{h_{ij}} - H_{ij}^t\}. \tag{4.1}$$

Replacing variable w_{ij}^t in equation 4.1, we have :

$$\min \sum_{[i,j]\in E} \sum_{t=0}^{T} P_{ij} w_{ij}^t \tag{4.2}$$

which denotes our first objective. The two constraints associated with (4.2) are $w_{ij}^t \geq 0$ and $w_{ij}^t \geq \overline{h_{ij}} - H_{ij}^t$.

Our second objective is a routing objective that aims to minimize dead-heading and

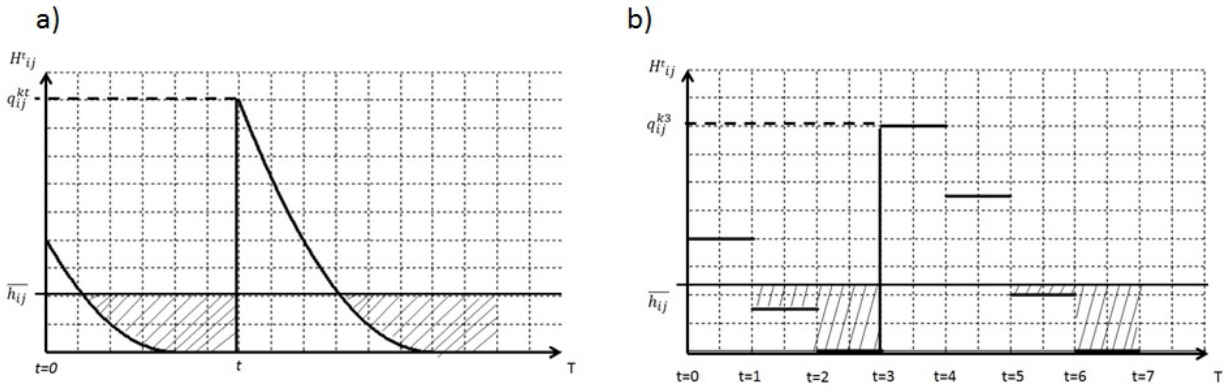

Figure 4.1 a) Humidity level of edge $[i, j]$ in a continuous time horizon. b) Humidity level of edge $[i, j]$ after the time horizon is discretized in equal periods.

watering costs. Because the direction in which roads are traversed is important for flow conservation, we use the variables associated with the arcs of the network. The complete model is shown below.

$$\min \sum_{[i,j]\in E} \sum_{t=0}^{T} P_{ij} w_{ij}^t + \sum_{(i,j)\in A} \sum_{k=1}^{K} \sum_{t=0}^{T} (c_{ij} X_{ij}^{kt} + r_{ij} Y_{ij}^{kt}) \tag{4.3}$$

s.t. :

$$w_{ij}^t \geq \overline{h_{ij}} - H_{ij}^t \qquad \forall [i,j] \in E,\ t \in \{0,\ldots,T\} \tag{4.4}$$

$$H_{ij}^{(t+1)} = (1 - (e_t b_{ij}))H_{ij}^t + \sum_{k=1}^{K} q_{ij}^{k(t+1)} \qquad \forall [i,j] \in E,\ t \in \{0,\ldots,T-1\} \tag{4.5}$$

$$H_{ij}^0 = H_{ij}^{int} \qquad \forall [i,j] \in E \tag{4.6}$$

$$q_{ij}^{kt} \leq H_{ij}^{max}(X_{uv}^{kt} + X_{vu}^{kt}) \qquad \forall \{[i,j] \in E | i = u, j = v; (u,v), (v,u) \in A\},\ t \in \{0,\ldots,T\},\ k \in \{1,\ldots,K\} \tag{4.7}$$

$$H_{ij}^t \leq H_{ij}^{max} \qquad \forall [i,j] \in E,\ t \in \{0,\ldots,T\} \tag{4.8}$$

$$\sum_{[i,j]\in E} q_{ij}^{kt} \leq Q^{kt} \qquad \forall t \in \{0,\ldots,T\},\ k \in \{1,\ldots,K\} \tag{4.9}$$

$$Q^{k0} = Q^{max} \qquad \forall\, k \in \{1,\ldots,K\} \tag{4.10}$$

$$Q^{kt} \leq Q^{max} \qquad \forall t \in \{0,\ldots,T\},\ k \in \{1,\ldots,K\} \tag{4.11}$$

$$Q^{k(t+1)} = Q^{kt} - \sum_{[i,j]\in E} q_{ij}^{kt} - q_{00}^{kt} \qquad \forall t \in \{0,\ldots,T-1\},\ k \in \{1,\ldots,K\} \tag{4.12}$$

$$q_{00}^{kt} \geq Q^{kt} - Q^{max} \qquad \forall t \in \{0,\ldots,T\},\ k \in \{1,\ldots,K\} \tag{4.13}$$

$$q_{00}^{kt} \geq -Q^{max} X_{00}^{kt} \qquad \forall t \in \{0,\ldots,T\},\ k \in \{1,\ldots,K\} \tag{4.14}$$

$$X_{ij}^{kt} + Y_{ij}^{kt} \leq 1 \qquad \forall (i,j) \in A,\ t \in \{0,\ldots,T\},\ k \in \{1,\ldots,K\} \tag{4.15}$$

$$\sum_{k=1}^{K} X_{ij}^{kt} \leq 1 \qquad \forall (i,j) \in A',\ t \in \{0,\ldots,T\} \tag{4.16}$$

$$\sum_{(i,j)\in A|i=0} X_{ij}^{k0} + Y_{ij}^{k0} = 1 \qquad \forall k \in \{1,\ldots,K\} \tag{4.17}$$

$$\sum_{(i,j)\in A|j=0} X_{ij}^{kT} + Y_{ij}^{kT} = 1 \qquad \forall k \in \{1,\ldots,K\} \tag{4.18}$$

$$X_{ij}^{kt} + Y_{ij}^{kt} \leq \sum_{l|(j,l)\in A} X_{jl}^{k(t+d_{ij})} + Y_{jl}^{k(t+d_{ij})} \qquad \forall (i,j) \in A,\ t \in \{0,\ldots,T\},\ k \in \{1,\ldots,K\} \tag{4.19}$$

$$X_{ij}^{kt} + Y_{ij}^{kt} \leq S_{ij}^{k(t+m)} \quad \forall (i,j) \in \Omega,\ m \in \{0,\ldots,d_{ij}-1\},\ t \in \{0,\ldots,T-m\},\ k \in \{1,\ldots,K\} \tag{4.20}$$

$$\sum_{(i,j)\in A'\setminus\{(u,v)\}} (X_{ij}^{kt} + Y_{ij}^{kt}) \leq 1 - S_{uv}^{kt} \quad \forall t \in \{0,\ldots,T\},\ (u,v) \in \Omega,\ k \in \{1,\ldots,K\} \tag{4.21}$$

$$H_{ij}^{t},\ w_{ij}^{t} \geq 0 \qquad \forall [i,j] \in E, t \in \{0,\ldots,T\} \tag{4.22}$$

$$q_{ij}^{kt} \geq 0 \qquad \forall [i,j] \in E, t \in \{0,\ldots,T\},\ k \in \{1,\ldots,K\} \tag{4.23}$$

$$q_{00}^{kt} \leq 0 \qquad \forall t \in \{0,\ldots,T\},\ k \in \{1,\ldots,K\} \tag{4.24}$$

$$X_{ij}^{kt},\ Y_{ij}^{kt} \in \{0,1\} \qquad \forall (i,j) \in A, t \in \{0,\ldots,T\},\ k \in \{1,\ldots,K\} \tag{4.25}$$

$$Q^{t} \geq 0 \qquad \forall t \in \{0,\ldots,T\} \tag{4.26}$$

$$S_{uv}^{kt} \in \{0,1\} \qquad \forall (u,v) \in \Omega, t \in \{0,\ldots,T\} \tag{4.27}$$

The objective function (4.3) minimizes both, the penalty cost assigned to the difference in humidity level and the routing cost for service and dead-heading. The penalized difference in humidity levels is either zero when $H_{ij}^{t} > \overline{h_{ij}}$ by constraints (4.22) or $\overline{h_{ij}} - H_{ij}^{t}$ by constraints (4.4). The humidity level at each time period is defined by (4.5). It increases with the quantity delivered by each of the water trucks and decreases with consumption rate, which is a function of the existing quantity. Constraints (4.6) indicate the initial humidity level. Constraints (4.7) ensure that the quantity of water delivered is less than the maximum humidity level of each edge. This set of constraints make the link between the variables related to edges and those related to arcs. Edge $[i,j]$ receives quantity q_{ij}^{kt} if it is serviced either in direction (i,j) or (j,i). Humidity level stays below the maximum humidity level, H_{ij}^{max}, by (4.8). The quantity of water delivered to each edge does not exceed the capacity of the truck by constraints (4.9). Constraints (4.10) set the initial water level of each of the trucks to the maximum capacity at the beginning of the time horizon. The set of constraints (4.11) ensure that the water level of the truck does not exceed its capacity. The set of constraints (4.12) indicates the water level of each truck in each time period. Two constraints are used for truck refilling, which takes place in an artificial arc $(0,0)$ placed at the depot. The quantity of water added to the truck, expressed by variable q_{00}^{kt}, is either the difference in water level by constraints (4.13), or the truck capacity by constraints (4.14), in case it is empty. Note that variable q_{00}^{kt} is negative by constraints (4.24) and it will be added to the capacity of the truck through constraints (4.12). The set of constraints (4.15) restrict a truck to traverse an arc with service or deadheading, but not both in the same time period. Constraints (4.16) limit the number of trucks that can service an arc to one in a given period of time. Constraints (4.17) and (4.18) allow the truck to start and end at the depot at the beginning and end of the time horizon respectively. Constraints (4.19) are the flow conservation constraints. They ensure that when a truck is traversing arc (i,j), either with service or deadheading, it will continue on an adjacent arc, (j,k), after d_{ij} time periods. Constraints (4.20) and (4.21) ensure that when a truck is traversing arc (i,j), either with service or deadheading, it stays there for d_{ij}

time periods. If variable $S_{uv}^{kt} = 1$ then by (4.21), no other arc can be traversed. To ilustrate the interaction between (4.20) and (4.21), consider the example for any truck k where the truck begins watering arc $(1, 2)$ at time 3, and $d_{12} = 3$ time periods, then $X_{12}^{k3} = 1$, making $S_{12}^{k3} = S_{12}^{k4} = S_{12}^{k5} = 1$ by (4.20). Then by (4.21), $X_{ij}^{kt} = Y_{ij}^{kt} = 0 \; \forall (i, j) \neq (1, 2), t = 3, 4, 5$. By (4.19), either $X_{2j}^{k6} = 1$ or $Y_{2j}^{k6} = 1$, making by (4.21) $S_{12}^{k6} = 0$ and by (4.20), $X_{12}^{kt} = Y_{12}^{kt} = 0$, for $t = 4, 5$, thus ensuring that only $X_{12}^{k3} = 1$ during periods 3, 4 and 5. Finally, the variables are defined in constraints (4.22) through (4.27).

This model was tested with a small network of 11 edges, for one to five trucks and for 20 to 25 time periods. Larger networks, with more trucks or more time periods result in an extremely large computation time. In order to solve larger instances for a longer time horizon, i.e., more than 25 time periods for the tested network, a heuristic algorithm is needed.

4.4 Adaptive large neighborhood search

The Adaptive Large Neighborhood Search (ALNS) was introduced by Ropke and Pisinger (Ropke et Pisinger, 2006) in 2006. It is based on the remove-insert heuristic proposed by Shaw (Shaw, 1997), in which the number of possible removal and insertion options results in a very large neighborhood to explore. Ropke and Pisinger combine several remove-insert heuristics at each iteration of the process selecting them based on their statistical performance. The ALNS has been applied to a similar node routing problem, the inventory routing problem (IRP) (Coelho *et al.*, 2012a), (Coelho *et al.*, 2012b), where a set of customers located on the nodes of a network require the delivery of a certain material and thus, inventory and routing decisions must be made simultaneously. In the arc routing problem domain, the ALNS has been applied to the synchronized arc routing problem (Salazar-Aguilar *et al.*, 2012), where a set of vehicles are coordinated for snow plowing operations on streets requiring simultaneous service by multiple vehicles. This algorithm has been used also for the synchronized arc and node routing problem (Salazar-Aguilar *et al.*, 2013) where a fleet of vehicles in charge of marking the streets of a network, coordinate to meet with a replenishment vehicle at specific intersections.

The proposed ALNS algorithm consists of two phases : An initial solution is obtained by means of a simple heuristic, and it is then modified in an improvement phase by means of a set of destroy and repair operators. The improvement process is divided into a number of *segments*. A segment consists of δ iterations. For our problem, we used 100 segments of $\delta = 250$ iterations each. In each one of these iterations, an operator is randomly selected using a *roulette-wheel* mechanism. The probability of selecting operator O_i is $\rho_i / \sum_{i=1}^{n} \rho_i$, where ρ_i

is the weight of operator i, defined as the ratio of the score of the operator C_i to the number of times it was used during the segment, U_i, i.e., $\rho_i = C_i/U_i$. The operators that performed better in the past have a higher probability of being selected. At the beginning of the algorithm, all the operators have the same weight, i.e. $\rho_i = 1/n$. When a new segment starts, the score of the operators is set to 0, and the weights are updated. Considering segment j, the weight of operator i in segment $j+1$ is given by :

$$\rho_{i,j+1} = \rho_{ij}(1-r) + r\frac{C_i}{U_i}$$

where $r \in \{0,1\}$ is the *reaction factor*. The weight is then a combination of its original value and its performance on the previous segment. Operators with a bad performance have a small probability of being selected, but not 0.

Three types of scores contribute to C_i : σ_1 is given if the operator delivers a better solution than the best solution found ; σ_2 is given to the operator if it delivers a better solution than the current one, but not better than the best solution, and it has not appeared before ; and σ_3 is given if the operator delivers a bad solution but is accepted with probability $e^{-(f'-f)/\tau}$, where f is the current solution, f' the new solution and τ, the temperature factor (Ropke et Pisinger, 2006). Note that $\sigma_1 > \sigma_2 > \sigma_3$. The temperature has an initial value, τ_0, calculated from the initial solution so that a solution that is $\mu\%$ worse than the current solution is accepted with 50% probability (Ropke et Pisinger, 2006). τ is decreased at every iteration by $\tau = \tau \times c$, where $0 < c < 1$ (Ropke et Pisinger, 2006). The value of c is calculated from τ_0 so that at the end of the 25000 iterations τ decreases to 0.01 (Coelho *et al.*, 2012b).

4.4.1 Initial solution

The initial solution is obtained by a *Cluster-First Route-Second* algorithm. The first step consists in partitioning the set of edges in K sets, where K is the number of trucks. We implemented the technique used by Monroy *et al.* (2011) to select K edges of the network, called *seeds*, as far apart from each other and from the depot. Once h seeds have been selected, $\{s_1, \ldots, s_h\}$, edge e is selected as seed s_{h+1} such that $dist_{0e}(\prod_{i=1}^{h} dist_{s_i e})$ is maximized, where $dist_{0e}$ is the lenght of the shortest path from edge e to the depot $(0,0)$ and $dist_{s_i e}$ is the length of the shortest path from e to seeds $s_1, \ldots, s_h$. The process is repeated until K seeds are obtained.

Once the seeds have been selected, the rest of the edges are assigned to one of these seeds by minimizing the sum of the shortest distances between them. Consider V_{ek} the lenght of the shortest path between edge e and seed k. Parameter D is the distance, in meters, tra-

veled in one time period at a constant truck speed, and d_e is the number of time periods for traversing edge e. D_G, the total network length obtained by adding the number of time periods needed to traverse the edges of the network, and multiplying it times parameter D i.e., $D_G = \sum_{e \in E} Dd_e$. π is a minimum percentage of the total network distance D_G. Variable $Z_{ek} = 1$ if edge e is assigned to seed k, and 0 otherwise.

$$Min \sum_{e \in E} \sum_{k \in K} V_{ek} Z_{ek} \tag{4.28}$$

s.t :

$$\sum_{k=1}^{|K|} Z_{ek} = 1 \qquad \forall e \in E \tag{4.29}$$

$$\sum_{e \in E} Dd_e Z_{ek} \geq \pi D_G \qquad \forall k \in K \tag{4.30}$$

$$Z_{ek} \in \{0, 1\} \qquad \forall e \in E,\ k \in K \tag{4.31}$$

The objective function (4.28) minimizes the sum of the distances between the edges and the selected seeds. Equation (4.29) ensures that all edges are assigned to one seed. Constraints (4.30) are used to balance the number of edges assigned to seeds. Variables are defined in the set of restrictions (4.31).

The result of the previous step is a set of lists $\{L_1, \ldots, L_k\}$ of available edges assigned to be serviced by each truck. It is important to notice that the edges in each one of these lists are in no particular order. They will eventually be serviced by a specific truck. We call a *run* a list of arcs in the order they will be serviced or traversed. The direction of traversal is important in a run, therefore, arcs are taken into account. The arcs in each of these runs should be ordered such that the depot appears in the beginning and end of it. At the end of each run, there is a refill process in which the truck stays at the depot for a determined number of periods. We call a *route*, a series of ordered runs and refill processes that are performed over the time horizon by a single truck.

The last step in the initial solution process is to create an initial route by means of a construction algorithm. For this process we consider a constant water rate α in liters per meter such that the quantity of water is a constant $Q = \alpha D d_{ij}$. λ_k is the set of *available edges* for truck k. An edge becomes available when its humidity level is such that it admits Q liters without exceeding its maximum humidity level. A solution is composed of four parts : f is the total cost, including routing and inventory costs ; R is the set of arcs that form the *route* ; Θ is the set of times at which the truck starts traversing each of the arcs in R ; and Φ is the set of quantities to be delivered to each of the arcs in R. Each one of these three sets

has the same number of elements and has a number of subsets, equal to the number of trucks. To illustrate how the solution looks, let us consider two trucks with $11000L$ capacity and 20 periods of time. Vehicle 1 will service the edges in $L_1 = \{[0,2],\ [2,3],\ [3,4],\ [3,5],\ [3,6]\}$, while truck 2 will service the edges in $L_2 = \{[0,3],\ [0,4],\ [0,5],\ [4,5],\ [4,7]\}$. Consider the following sets :

$R = \{\{(0,2),\ (2,3),\ (3,6),\ (6,3),\ (3,4),\ (4,3),\ (3,2),\ (2,0),\ (0,0)\},$
$\{(0,3),\ (3,0),\ (0,4),\ (4,5),\ (5,0)\}\}$;
$\Theta = \{\{1,2,4,9,13,15,17,19,20\},\ \{1,6,11,14,18\}\}$;
$\Phi = \{\{1500,0,4600,0,2300,0,2100,0,0\},\ \{4500,0,3200,3000,0\}\}$;

We interpret the solution as follows : Vehicle 1 starts watering edge $[0,2]$, in direction $(0,2)$, at the beginning of time 1, delivering $1500L$. Because it takes one time period to serve it, the truck starts traversing arc $(2,3)$ at the beginning of time 2. The quantity delivered is 0, in other words, no service is provided to edge $[2,3]$. Arc $(2,3)$ takes two time periods to be traversed, therefore, at the beginning of time 4 the truck starts watering edge $[3,6]$, by traversing it in direction $(3,6)$ delivering $4600L$. The run continues until the truck arrives to the depot at time 19 and remains there at time 20. Note that truck 1 delivers $2100L$ to edge $[2,3]$, by traversing in direction $(3,2)$ at the beginning of time 17. Because of the truck's limited capacity, only four edges were serviced in this run, leaving edge $[3,5]$ unserviced. A total of $10500L$ of the $11000L$ available were used for this run. The remaining $500L$ do not correspond to the constant amount to be delivered to edge $[3,5]$. This means in the next run, $[3,5]$ will be selected for service by the first truck. The second subset in R shows the arcs traversed or watered by truck 2, the second subsets in Θ and Φ show the corresponding times and quantities to each one of the edges.

To form the initial run, the edge with the highest priority is selected first, followed by adjacent edges regardless of their priority. The process of selection continues until the truck capacity is reached. This process is described by Algorithm 1.

Once the edges have been selected, a complete run is formed using the selected edges, and the shortest path between them and the depot. This step is shown in Algorithm 2. The number of periods required for the run, as well as the quantities used are updated and a solution is then formed. The runs become elements of R. The time periods and quantity of water associated with each arc in R, become elements of sets Θ and Φ respectively. The process is repeated until the time horizon T is reached.

Routing and inventory costs are calculated for every time period in the time horizon, including those periods when the truck is traversing/watering arcs and when the truck is refilling at the depot. This process is shown in Algorithm 3. It requires a partial solution (R_g, Θ_g, Φ_g). Routing costs c_{ij} and r_{ij} are the first to be calculated with the information

Algorithm 1 EDGE SELECTION

Input : λ_k : Set of available edges ; α : The water rate ; D : Distance traversed in one time period.
Edge parameters : d_{ij}, P_{ij}, for all $[i, j] \in E$.
Truck parameters : Q^{max}
Output : EDGES : List of edges to be serviced.
Initialize : EDGES $= \emptyset$; $\lambda = \lambda_k$
selected_edge = Edge with a maximum P_{ij} from λ
$Q = \alpha D d_{ij}$ quantity delivered to selected_edge
Neighbors $= \emptyset$: Set of adjacent edges to selected_edge
while $Q < Q^{max}$ **do**
  Remove selected_edge from λ
  Add selected_edge to EDGES
  if $\lambda = \emptyset$ **then**
    Break
  else
    for x in EDGES **do**
      for y in λ **do**
        if y is adjacent to x **then**
          Add y to Neighbors
        end if
      end for
    end for
    if Neighbors $= \emptyset$ **then**
      Find the closest edge in λ and add it to Neighbors.
    end if
    selected_edge = first element in Neighbors
    $Q = Q + \alpha D d_{ij}$
  end if
end while
return EDGES

provided by the arcs in R_g and their corresponding quantities.

In order to find inventory costs, first we need to find Δ_1, the difference between the humidity level and the desirable humidity level $\overline{h_{ij}}$ at each time period. If this difference is positive, the inventory cost is calculated. Set λ_k is also updated at each time period, so that the edges can be selected from it when the next run is formed. An edge $[i, j]$ becomes element of λ_k when Δ_2, the difference between the maximum capacity and the humidity level, is such that over-watering is avoided. This process is shown in Algorithm 3.

Algorithm 2 INITIAL SOLUTION

Input : $G = (N, A)$; T : Number of periods in the time horizon ; K : number of trucks ; L_k ;
refill_time : time needed to refill the truck
Edge parameters : I_{ij}, d_{ij}, for all $[i, j] \in E$.
Truck parameters : Q^{max}
Output : (f, R, Θ, Φ)
Initialize : $f = 0$; $R = \Theta = \Phi = \emptyset$; $H_{ij} = Iij$, for all $[i, j] \in E$.
for k in K **do**
$\lambda_k = L_k$: Set of available edges for truck k ; $f_k = 0$; $R_k = \Theta_k = \Phi_k = \emptyset$; $t = 0$.
while $t \leq T$ **do**
$R_g = \Theta_g = \Phi_g = \emptyset$: Partial solution includes sets of routes, times and quantities.
EDGES = Run EDGE SELECTION
ARCS = Route from the depot to EDGES and back to the depot.
for a in ARCS **do**
Add a to R_g
t_a = time at which truck k starts traversing a.
q_a = quantity of water delivered to arc a.
Add t_a to Θ_g and q_a to Φ_g.
end for
current_time = $t_a + d_a$, number of time periods required to traverse the last element of R_g
add $(0, 0)$ to R_g ; current_time to Θ_g ; 0 to Φ_g
current_time = t_a+refill_time
if current_time $> T$ **then**
Arrange R_g, Θ_g, Φ_g to end at T.
COST, H_{ij}, λ_k = Run FIND COSTS using R_g, Θ_g and Φ_g.
else
COST, H_{ij}, λ_k = Run FIND COSTS using R_g, Θ_g and Φ_g.
end if
$f_k = f_k$+COST
Add the elements of R_g to R_k, Θ_g to Θ_k and Φ_g to Φ_k
$t = t + current\_time$
end while
$f = f + f_k$; add R_k to R, Θ_k to Θ, Φ_k to Φ
end for
return (f, R, Θ, Φ)

Algorithm 3 FIND COSTS

Input : current_time ; λ_k ; e_t ; H_{ij}, for all $[i,j] \in E$: current inventory level ; R_g, Θ_g, Φ_g
Edge parameters : $\overline{h_{ij}}$, P_{ij}, b_{ij}, r_{ij}, c_{ij}, for all $[i,j] \in E$.
Output : COST, H_{ij}, λ_k
Initialize : TIME=first element of Θ, COST=0
while TIME $\leq$ current_time **do**
if TIME in Θ_g **then**
arc_{ij} = The arc in R_g that corresponds to TIME.
$quantity_{ij}$ = The quantity in Φ_g that corresponds to TIME.
for h in H_{ij} **do**
if $[i,j]$ corresponds to arc_{ij} **then**
$h = h + quantity_{ij}$
end if
end for
Find the traversing cost r_{ij} and watering cost c_{ij} for arc_{ij}.
route_cost = $c_{ij} + r_{ij}$
else
route_cost = 0
end if
penalty_cost = 0
for h in H_{ij} **do**
Find water consumption $(b_h + e_{TIME})$
$h = h \times (1 - (b_h + e_{TIME}))$
$\Delta_1 = \overline{h_{ij}} - h$
if $\Delta_1 < 0$ **then**
inventory_cost=0
else
inventory_cost= $P_{ij} \times \Delta_1$
end if
$\Delta_2 = H_{ij}^{max} - h$
if Δ_2 allows watering **then**
Add $[i,j]$ to λ_k
end if
penalty_cost = penalty_cost + inventory_cost
end for
COST = COST + penalty_cost + route_cost
TIME = TIME + 1
end while
return COST, H_{ij}, λ_k

4.4.2 Improvement phase

Once the initial solution has been obtained, we continue to the improvement phase described in Algorithm 4.

Algorithm 4 Improvement phase for the ALNS

Input : (f, R, Θ, Φ) : Initial solution
$\sigma_1, \sigma_2, \sigma_3, \mu, r, \{O_1, \ldots, O_n\}$
Output : $(f^*, R^*, \Theta^*, \Phi^*)$: Best solution Initialize $(f^*, R^*, \Theta^*, \Phi^*) = (f, R, \Theta, \Phi)$, $(f'', R'', \Theta'', \Phi'') = (f, R, \Theta, \Phi)$: Current solution, $U_i = 0, C_i = 0, \rho_i = 1/n$, for all $i \in \{O_1, \ldots, O_n\}$
$segment = 1$
$\tau = \tau_0$, from $e^{-(f'-f)/\tau} = 0.5$, where $f' = (1+\mu)f$
Compute c from the initial solution so that $\tau = 0.01$ at the end of 25000 iterations.
while $segment \leq 100$ **do**
  while $\delta \leq 250$ **do**
    Choose an operator from $\{O_1, \ldots, O_n\}$ using the roulette wheel selection mechanism. A solution (f', R', Θ', Φ') is obtained.
    if $(f', R', \Theta', \Phi') < (f^*, R^*, \Theta^*, \Phi^*)$ **then**
      $(f^*, R^*, \Theta^*, \Phi^*) = (f', R', \Theta', \Phi')$, $U_i = U_i + 1$, $C_i = C_i + \sigma_1$
    else if $(f', R', \Theta', \Phi') < (f'', R'', \Theta'', \Phi'')$ and the solution has not appeared **then**
      $(f'', R'', \Theta'', \Phi'') = (f', R', \Theta', \Phi')$, $U_i = U_i + 1$, $C_i = C_i + \sigma_2$
    else if *random number* $< e^{-(f'-f'')/\tau}$ **then**
      $(f'', R'', \Theta'', \Phi'') = (f', R', \Theta', \Phi')$, $U_i = U_i + 1$, $C_i = C_i + \sigma_3$
    else
      $U_i = U_i + 1$
    end if
    $\tau = \tau \times c$, $\delta = \delta + 1$
  end while
  $segment = segment + 1$
  $\rho_i = \rho_i(1-r) + r \times C_i/U_i$, for all $i \in \{O_1, \ldots, O_n\}$
  $U_i = 0, C_i = 0$ for all $i \in \{O_1, \ldots, O_n\}$
end while
return $(f^*, R^*, \Theta^*, \Phi^*)$

We developed eight destroy/repair operators for the case with constant watering rate :

O_1. *Single edge exchange.* Exchanges one edge, randomly selected from two different lists L_k and L_l, $k \neq l$.

O_2. *Adjacent edges exchange.* Exchanges two adjacent edges, randomly selected, from two different lists L_k and L_l, $k \neq l$.

O_3. *Multiple edges exchange.* Exchanges a random number of edges from two different

lists L_k and L_l, $k \neq l$.

O_4. *Service status change*. Exchanges the status of randomly selected edges from *serviced* to *not serviced* and vice versa. The total quantity of water used to spray the edges must be less than Q^{max} so that the solution remains feasible. The operation applies to all available edges other than the edge with the highest priority, which keeps its status.

O_5. *Same truck sequence exchange*. Two sequences of arcs (serviced or not) are chosen from two different, randomly chosen runs performed by the same truck. The number of arcs in the sequence as well as its position inside the run is chosen at random. Both sequences are exchanged. Because the inserted arcs may not be adjacent to the existing arcs in the run, the shortest path between them is calculated. To prevent an infeasible solution, the capacity of the truck is checked after the insertion takes place. If the quantity of water exceeds Q^{max}, the last arcs of the run are removed one by one until feasibility is restored and the last arc is connected to the depot. One of the runs with the inserted arcs becomes the first run. To prevent non available arcs to be inserted in a later route, the rest of the solution is recalculated.

O_6. *Different truck sequence exchange*. This operator is similar to O_5 except that the exchange occurs with two sequences performed by different trucks. The sequences are extracted and inserted in the first run of each truck in order to avoid non available arcs to be inserted at later runs. The subsequent runs are obtained from them. The mechanism used in operator O_5 to prevent exceeding the capacity of the truck is implemented in O_6 as well.

Because insertions in O_5 and O_6 can lead to re-watering of the same edge, Δ_2 is checked at every time period. If the difference, $\Delta_2 \leq 0$, the insertion becomes unfeasible and is rejected.

O_7. *Run exchange*. This operator changes the order in which two runs are performed by the same truck.

O_8. *Watering rate change*. This operator assigns a random watering rate to each edge, thus changing the quantity of water delivered. The initial solution has a constant rate of $\alpha = 3.6L/m$ that ensures a non-random value used for later comparison. Operator O_8 changes these rate between two values $(\alpha_{min}, \alpha_{max})$ with the restriction that the selected rate does not result in a delivered quantity that exceeds H_{ij}^{max}. The main purpose of O_8 is to assign a different value to variable q_{ij}^{kt}.

Operators O_1, O_2, O_3 increase diversification because they modify the lists of trucks before a route is formed. The rest of the operators increase intensification. Instead of working with a set of *destroy* operators and another set of *repair* operators, each one of them destroys the solution and immediately repairs it.

4.4.3 The problem of inventory constraints

A solution that is destroyed and repaired by each one of the operators requires that the humidity levels of every edge and all the costs be calculated again for every time period. The reason for this is that equation (4.5) carries the humidity level from one time period to the next, and because the costs are calculated based on the humidity level for each time period, both need to be computed again regardless of how small the change in the initial solution is. In other words, the process described in Algorithm 3 has to be done for every time period in the time horizon. This constitutes a disadvantage in terms of computation time because the smallest change in any component of the solution, e.g., watering times, quantity of water delivered, edge sequence, requires the rest of the solution to be rebuilt.

Consider, for example, two edges A and B. The initial solution shows that edge A will be watered at the beginning of period t_1, before edge B, that will be watered at the beginning of t_2. If an operator changes the order so that B is watered at the beginning of t_1, it does not mean that A will be watered at the beginning of t_2 due to the fact that the path $A - B$ may be different from $B - A$. Moreover, the costs associated with not watering edge A at t_1 need to be calculated again as do the inventory levels for the rest of the time periods.

4.5 Computational results

No benchmark instances are available for this problem. We used 22 of the 23 *gdb* instances for CARP Golden *et al.* (1983) that are available at http ://www.uv.es/belengue/carp.html. These instances were adapted to meet the characteristics of our problem and are labeled *gdbj*. Instance *gdbj13* was not tested because the length of the edges of the network required more than 300 time periods in the time horizon to complete a significant run. We also developed 10 instances based on five real open-pit mine networks that were designed to be tested with different number of trucks. Instances labeled *mineA* simulate 3 trucks, while instances labeled *mineB* simulate the use of 5 trucks. The *mine* networks were obtained by observing satellite images such as the one depicted in Figure 4.2, that can be found in http ://maps.google.com with coordinates (-22.22666, -68.866695). The distance was calculated using Google maps.

The time horizon is set to $T = 300$ time periods. Some adjustments were made to these instances. The traversing cost, r_{ij}, is related to the arc distance, (d_{ij}), while the watering cost is set to $c_{ij} = r_{ij} + 2$. The penalty cost, P_{ij}, is set to be at least as the maximum watering cost, i.e., $\min P_{ij} \geq \max c_{ij}$, so that penalty costs for not watering have priority in the objective function. The required level of humidity is set to $\overline{h_{ij}} = \alpha D d_{ij}$. H_{ij}^{int} and H_{ij}^{max} are set to

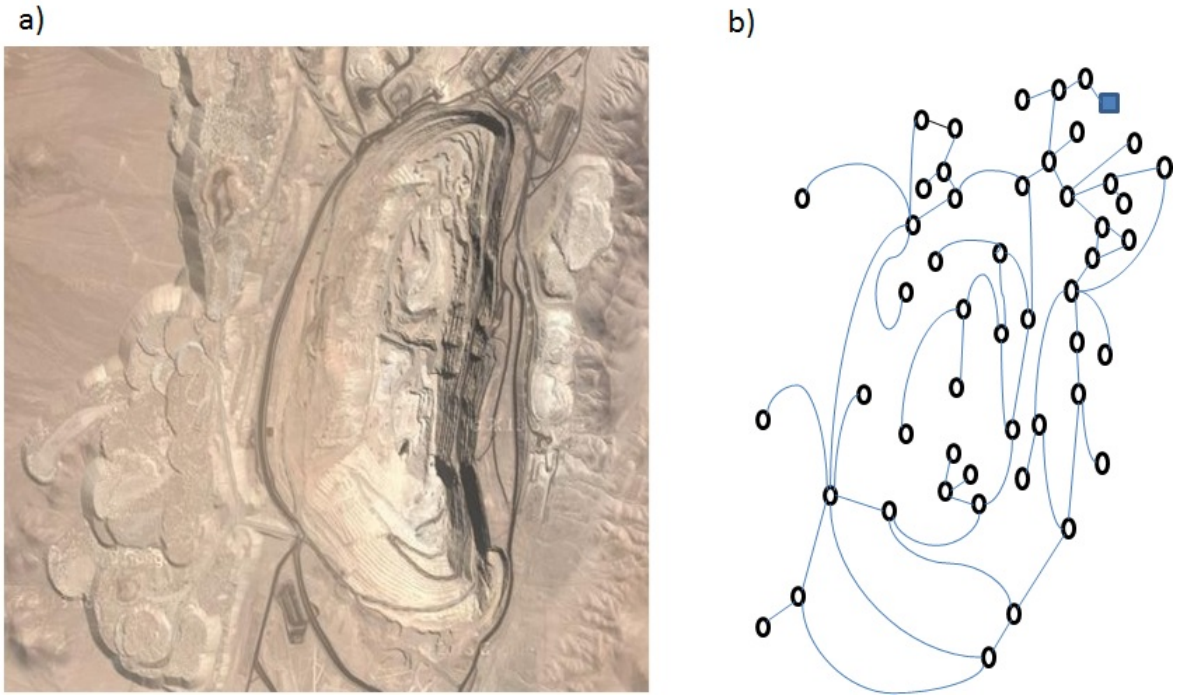

Figure 4.2 a. Satellite view of one of the mines used to determine instances *mine*. b. The network that correponds to the pit shown in a.

$0.1\overline{h_{ij}}$ and $1.1\overline{h_{ij}}$ respectively. b_{ij} is set to 3% for low priority edges and 10% to high priority ones. $e_t = e_{t-1} + 0.1$ if $\lceil 0.55T \rceil \leq t < \lceil 0.75T \rceil$; $e_t = e_{t-1} - 0.1$ if $\lceil 0.75T \rceil \leq t < \lceil 0.95T \rceil$; and $e_t = 1$ otherwise. These parameters are used to approximate the evaporation rate during daytime, which is increased on the second half of the time horizon as shown in the results of hourly evaporation fund in Molina-Martínez *et al.* (2006).

The algorithm was coded in Python and executed on a 2.27 GHz Intel Core i3 Notebook PC.

4.5.1 Individual performance of the operators

In order to build a basis for comparison, we tested the eight operators separately for each one of the *gdbj* and *mine* instances. Figure 4.3 shows the percentage of improvement of the initial cost for *gdbj* instances while Figure 4.4 shows the percentage of improvement of the initial cost for *mine* instances. Table 4.1 shows the results of testing the operators separately for *gdbj* instances. The results for *mine* instances are shown in Table 4.2. Columns 2 and 3 in these tables show the number of nodes and edges respectively for each network. Column 4

shows the cost of the initial solution. Column 5 shows the cost of the best solution found. All objective function values were divided by 1×10^6. Columns 6 and 7 show the percentage of improvement of the initial solution and the running time in seconds respectively. The tests stop when 5000 iterations are reached or the running time reaches 1080 seconds. The operator that performed best is shown in the last column.

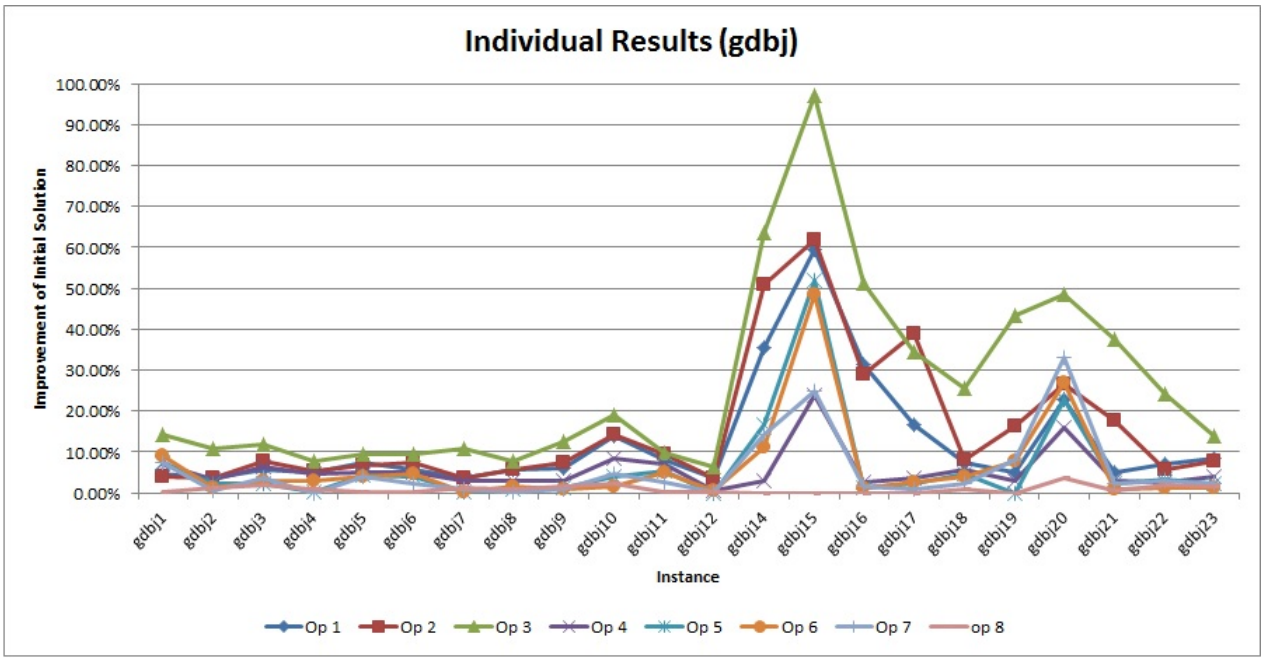

Figure 4.3 Percentage of improvement of the cost of the initial solution when using the eight operators separately for *gdbj* instances.

When the operators are tested separately, 21 of the 22 *gdbj* instances and 5 of the 10 *mine* instances show that operator O_3 provides the best cost improvement. Moreover, it is the diversification operators (O_1, O_2, O_3) that perform better than the rest, O_1 in *mine1B* and *mine4B* instances and O_2 in instances *gdbj17, mine3B, mine4A* and *mine5A*.

A second important remark is the difference in the initial solution between operators *mineA* and *mineB* in column four of Table 4.2. For each one of the five different networks, the instance with 5 trucks (B) results in a better initial solution than the one with 3 trucks (A). This result was expected considering that more trucks are able to water more edges, and it works as long as investment costs are not considered.

The individual performance of intensification operators, i.e., O_4, O_5, O_6, O_7 and O_8, suggests that in order for them to achieve a significant improvement they need to be combined with diversification operators, i.e., O_1, O_2 and O_3. This is our basis to select the combinations

Tableau 4.1 Best solution from running the eight operators separately for *gdbj* instances.

Network	Size		Cost of the Initial Solution	Cost of the Best Solution	Improvement %	Computation Time (sec)	Operator
	N	E					
gdbj1	12	22	736.7	631.7	14.26%	343.9	O_3
gdbj2	12	26	862.9	770.7	10.69%	414.4	O_3
gdbj3	12	22	669.8	589.5	11.99%	378.5	O_3
gdbj4	11	19	697.3	643.3	7.74%	303.5	O_3
gdbj5	13	26	1101.8	997.9	9.43%	383.4	O_3
gdbj6	12	22	850.8	771.3	9.34%	355.2	O_3
gdbj7	12	22	771.3	686.7	10.96%	348.4	O_3
gdbj8	27	46	1513.7	1393.9	7.91%	1080.2	O_3
gdbj9	27	51	1338.3	1168.9	12.66%	1080.0	O_3
gdbj10	12	25	996.9	808.0	18.94%	376.2	O_3
gdbj11	22	45	1074.7	969.0	9.83%	796.8	O_3
gdbj12	13	23	15427.0	14432.3	6.45%	330.2	O_3
gdbj14	7	21	382.0	139.5	63.47%	386.7	O_3
gdbj15	7	21	166.4	4.7	97.19%	479.3	O_3
gdbj16	8	28	576.7	280.3	51.40%	453.0	O_3
gdbj17	8	28	420.8	257.2	38.89%	492.8	O_2
gdbj18	9	36	779.3	580.5	25.51%	574.0	O_3
gdbj19	8	11	81.9	46.4	43.35%	288.8	O_3
gdbj20	11	22	602.0	309.6	48.57%	385.4	O_3
gdbj21	11	33	825.8	516.5	37.45%	649.3	O_3
gdbj22	11	44	1313.7	993.5	24.37%	849.7	O_3
gdbj23	11	55	1297.8	1115.3	14.06%	1014.6	O_3

Tableau 4.2 Best solution from running the eight operators separately for *mine* instances.

Network	Size		Cost of the Initial Solution	Cost of the Best Solution	Improvement %	Computation Time (sec)	Operator
	N	E					
mine1 A	21	22	1329.2	1272.4	4.28%	557.0	O_3
mine1 B	21	22	1248.3	1174.9	5.88%	757.1	O_1
mine2 A	22	27	3269.7	3126.8	4.37%	517.7	O_3
mine2 B	22	27	3252.2	3029.5	6.85%	646.1	O_3
mine3 A	49	53	5812.8	5666.4	2.52%	1080.4	O_3
mine3 B	49	53	5707.1	5519.7	3.29%	1080.4	O_2
mine4 A	51	60	4183.2	4046.9	3.26%	1080.1	O_2
mine4 B	51	60	4065.5	3938.6	3.12%	1081.5	O_1
mine5 A	30	35	2494.7	2396.9	3.92%	791.0	O_2
mine5 B	30	35	2428.6	2308.0	4.97%	1080.1	O_3

of operators that will be tested in the next section.

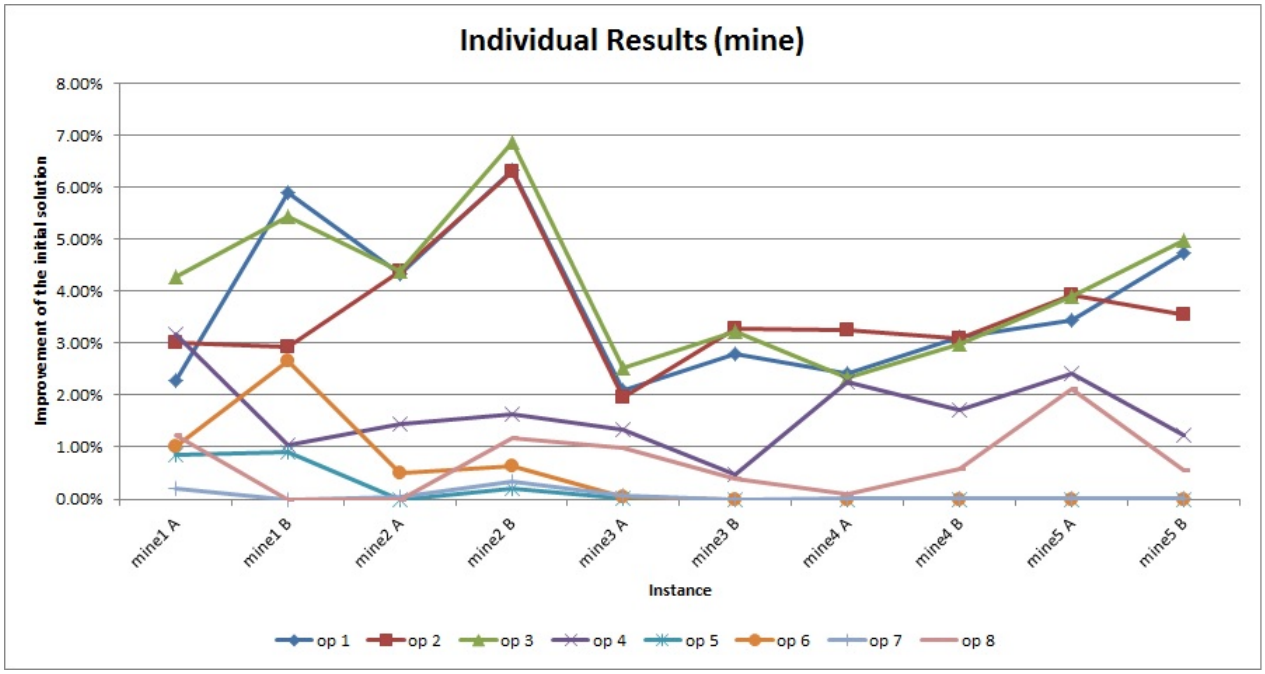

Figure 4.4 Percentage of improvement of the cost of the initial solution when using the eight operators separately for *mine* instances.

4.5.2 Parameter tuning

Before starting the tests we performed a tuning phase for the ALNS parameters involved in the selection of operators, i.e., r, $\sigma_1, \sigma_2, \sigma_3$ and μ Ropke et Pisinger (2006). No parameters are involved in the operators because they perform random operations. For the tuning process we ran the ALNS program with only 1000 iterations for each one of the instances and then repeated the run three times. We started testing different values of the first parameter while keeping the other parameters fixed. The value of the parameter that delivered the best average improvement, compared to the best solution available, of the 5 runs was selected. Keeping this parameter fixed we selected different values for the next one and repeated the process until all the parameters were tested. In the case of σ_1, σ_2 and σ_3, after selecting σ_1, we selected different combinations of the other two scores between 1 and σ_1. The selected parameters $(r, \mu, \sigma_1, \sigma_2, \sigma_3)$ are (0.8, 0.5, 15, 10, 5) for *gdbj* instances and (0.1, 0.8, 25, 10, 5) for *mine* instances.

4.5.3 Combinations of operators

We tried different combinations of operators to observe the interaction among them. We include operators O_1, O_2, and O_3 in every combination because they determine the diversification of the process. We decided to include also, operator O_8 in every combination because it is the only one that changes the quantity of water delivered and thus, determines q_{ij}^{kt}. These four operators are called *basic* and determine the base of each combination. The selected combinations are shown in Table 4.3 for *gdbj* instances and Table 4.4 for *mine* instances. The two columns of each combination show the percentage of improvement with respect to the value of the objective function of the initial solution and the the computation time. The stopping criterion is 25000 iterations or 7200 seconds.

Tableau 4.3 Percentage of improvement on the initial solution and computation time for the combinations of operators used for *gdbj* instances.

Network	Base		Base, O_4		Base, O_5		Base, O_6		Base, O_7		Base, O_4, O_5		Base, O_4, O_5, O_6		All	
	Imp. %	Time (sec)	Imp. %	Time (sec)	Imp. %	Time (sec)	Imp. %	Time (sec)	Imp. %	Time (sec)	Imp. %	Time (sec)	Imp. %	Time (sec)	Imp. %	Time (sec)
gdbj1	**16.99%**	1599.1	10.98%	1583.1	13.51%	1658.3	13.48%	1592.3	11.05%	1550.4	12.46%	1610.7	9.94%	1543.9	6.56%	1592.9
gdbj2	9.85%	1882.6	12.30%	1857.2	10.13%	1841.7	**12.65%**	1862.7	11.41%	1851.0	9.08%	1843.8	10.47%	1854.4	7.73%	1827.6
gdbj3	**13.87%**	1583.6	11.84%	1635.6	12.30%	1583.8	11.84%	1577.7	13.49%	1559.7	11.96%	1574.2	12.86%	1533.1	13.22%	1619.9
gdbj4	**13.77%**	1404.9	6.73%	1346.7	4.94%	1501.7	5.79%	1416.0	6.12%	1285.2	5.67%	1494.0	5.16%	1513.2	13.61%	1264.3
gdbj5	11.59%	1843.2	12.40%	1798.7	12.28%	1869.7	**13.56%**	1862.2	7.67%	1842.8	10.86%	1852.7	10.12%	1794.5	8.21%	1756.3
gdbj6	12.16%	1661.1	15.64%	1559.2	13.99%	1605.0	**18.22%**	1640.1	9.89%	1665.2	16.45%	1601.9	13.23%	1593.5	10.92%	1493.0
gdbj7	10.07%	1735.4	4.59%	1589.4	13.18%	1533.2	**14.02%**	1550.1	13.25%	1495.3	8.71%	1606.0	5.16%	1522.7	5.60%	1521.5
gdbj8	5.97%	4458.2	6.90%	4985.4	7.63%	5609.7	7.63%	5609.7	**8.53%**	5394.1	7.64%	5528.9	7.19%	4960.5	6.06%	5869.0
gdbj9	15.70%	6569.7	13.16%	5238.1	19.03%	6694.5	**22.84%**	6371.0	12.21%	5948.3	12.06%	7200.7	12.84%	6843.6	16.08%	5632.5
gdbj10	16.60%	1758.5	20.88%	1822.0	18.81%	1950.9	**20.63%**	1779.8	20.54%	1904.6	18.40%	1998.5	17.47%	1728.8	15.92%	1792.5
gdbj11	8.81%	3927.1	9.27%	4286.1	10.53%	3756.5	8.80%	4022.0	**10.85%**	3568.4	8.04%	4129.9	6.53%	4409.5	10.15%	3664.7
gdbj12	6.32%	1594.2	6.47%	1600.0	6.08%	1565.7	6.28%	1617.4	**6.56%**	1631.9	6.08%	1574.5	6.39%	1595.1	6.08%	1675.2
gdbj14	61.55%	1732.1	**66.07%**	1662.4	64.33%	1724.4	51.11%	1776.1	61.55%	1743.2	45.51%	1897.8	49.52%	1811.3	54.58%	1720.5
gdbj15	70.83%	2469.6	73.19%	2192.9	88.04%	2350.1	96.99%	2335.1	**99.11%**	2352.7	79.12%	2229.4	74.75%	2295.6	55.99%	2295.0
gdbj16	44.76%	2515.7	43.15%	2421.6	28.71%	2608.4	**50.46%**	2279.1	38.98%	2366.0	40.73%	2284.5	40.82%	2220.4	32.78%	2268.2
gdbj17	32.44%	2664.1	34.89%	2630.1	27.16%	2934.8	50.48%	2589.7	**55.53%**	2763.2	35.42%	2569.4	29.40%	2798.4	40.93%	2624.3
gdbj18	32.58%	2819.1	**33.60%**	2787.5	17.41%	2833.1	18.90%	2776.0	26.83%	2760.2	27.32%	2929.3	24.75%	2921.5	27.05%	2723.5
gdbj19	96.89%	1413.2	96.89%	1337.3	96.89%	1490.6	96.89%	1126.4	97.35%	1372.4	96.89%	1383.2	96.89%	1401.6	**97.58%**	1333.1
gdbj20	43.13%	2080.5	36.79%	1982.0	42.77%	2088.8	**46.20%**	1950.6	39.21%	2006.6	48.48%	1879.5	40.69%	1844.0	27.91%	2132.4
gdbj21	33.39%	2867.7	25.86%	2987.1	40.05%	2618.3	**37.19%**	2748.3	28.81%	2725.9	32.03%	2798.8	35.39%	2711.7	33.37%	2873.2
gdbj22	**27.75%**	4093.7	17.12%	4003.6	17.65%	4294.8	16.40%	3706.2	22.24%	4041.3	14.46%	3756.5	13.24%	3767.4	22.86%	4051.0
gdbj23	24.13%	5055.2	17.03%	5051.1	14.81%	4873.7	16.78%	5395.9	13.43%	4893.3	13.49%	4874.8	16.74%	4849.5	7.54%	4655.2

It is important to notice that larger networks such as *gdbj9, mine3* and *mine4* reach the stopping criterion of 7200 seconds. This helped us establish that the size of the networks that the algorithm is able to solve within two hours is between 40 to 50 edges.

The best percentage of improvement is highlighted for each instance in both Tables 4.3 and 4.4. There is not enough evidence to conclude that a certain combination performs better than the others or that one combination dominates the others in terms of improvement of the initial value of the objective function. In Table 4.4, for example, four of the ten instances

Tableau 4.4 Percentage of improvement on the initial solution and computation time for the combinations of operators used for *mine* instances.

Network	Base		Base, O_4		Base, O_5		Base, O_6		Base, O_7		Base, O_4, O_5		Base, O_4, O_5, O_6		All	
	Imp. %	Time (sec)	Imp. %	Time (sec)	Imp. %	Time (sec)	Imp. %	Time (sec)	Imp. %	Time (sec)	Imp. %	Time (sec)	Imp. %	Time (sec)	Imp. %	Time (sec)
mine1 A	2.45%	2410.7	2.29%	2475.9	3.12%	2683.1	2.65%	2746.0	3.22%	2704.4	3.23%	2717.6	2.65%	2568.9	**5.48%**	2425.5
mine1 B	5.77%	3911.4	6.00%	3420.0	5.68%	3415.4	5.98%	3444.6	5.77%	3073.1	5.35%	3138.4	5.69%	3374.7	**6.10%**	3342.1
mine2 A	4.59%	2546.1	4.59%	2623.6	3.74%	3235.4	**5.23%**	2497.7	4.67%	2099.1	4.91%	2506.4	4.72%	2530.5	3.99%	2285.8
mine2 B	6.68%	3842.2	7.02%	3748.4	**7.29%**	4066.1	7.11%	3595.1	6.13%	3363.6	6.95%	4205.9	6.16%	3539.5	7.22%	3575.7
mine3 A	3.28%	7200.5	1.92%	7200.6	**3.83%**	7200.9	3.17%	7200.5	2.81%	7200.4	2.79%	7200.8	2.97%	7200.5	2.73%	7200.4
mine3 B	0.52%	7202.6	1.11%	7201.9	**1.43%**	7202.5	1.42%	7202.8	1.26%	7202.3	0.47%	7203.3	0.65%	7201.8	0.00%	7202.9
mine4 A	0.00%	7201.3	0.02%	7200.8	0.00%	7201.1	**0.40%**	7201.3	0.17%	7201.4	0.00%	7201.2	0.00%	7202.0	0.06%	7200.8
mine4 B	0.00%	7202.0	**0.04%**	7203.6	0.00%	7202.3	0.00%	7202.0	0.00%	7202.5	0.00%	7202.2	0.00%	7202.4	0.00%	7202.7
mine5 A	3.96%	5880.8	4.68%	4291.7	**5.67%**	5093.5	4.27%	3603.2	4.49%	3772.2	3.68%	3715.9	5.46%	3432.2	5.07%	4201.3
mine5 B	5.13%	5898.5	5.90%	7200.4	5.84%	7200.5	6.10%	7200.3	4.63%	5595.3	**6.20%**	7200.5	5.70%	7200.4	5.25%	6424.2

resulted in a higher improvement when combination *Base* and O_5 was used. However we cannot conclude that it is the best combination. We then compared the average performance of the operators to the best available value of the objective function, shown in column 6 of Tables 4.1 and 4.2 of section 4.5.1. The results of this comparison are shown on Table 4.5 for the *gdbj* instances and Table 4.6 for *mine* instances. The negative percentages mean that the improvement obtained by that combination is less than the best improvement obtained by an individual operator.

On average, the combination of base operators (O_1, O_2, O_3, O_8) with operator O_6 performs better for the set of *gdbj* instances, while the combination of the base operators with operator O_5 performs better for *mine* instances.

4.5.4 Networks with low percentage of improvement

A general comparison made between Tables 4.3 and 4.4 shows that the percentage of improvement of the initial value of the objective function for the *mine* instances, which correspond to road networks of real mines, is less than 10% in every case, as opposed to *gdbj* instances, where, in some cases, this percentage reaches more than 95%. A possible cause for this behavior is the shape of the *mine* instances, which resembles a tree. The truck traveling from the depot has a limited number of paths available to choose in order to reach the more distant edges in the network. Another possible cause is the truck's capacity. The larger its capacity, the more edges it will be able to water before going back to the depot for refill. To test the effect of truck capacity, we compared three different scenarios : a restricted truck capacity which allows it to water only a few edges per route ; the regular truck capacity,

Tableau 4.5 Percentage of improvement of the combinations against the best solution for *gdbj* instances.

Network	Best Improvement	Base	Base, O_4	Base, O_5	Base, O_6	Base, O_7	Base, O_4, O_5	Base, O_4, O_5, O_6	All
gdbj1	14.26%	2.73%	-3.28%	-0.74%	-0.78%	-3.20%	-1.79%	-4.31%	-7.70%
gdbj2	10.69%	-0.84%	1.60%	-0.57%	1.95%	0.71%	-1.61%	-0.22%	-2.96%
gdbj3	11.99%	1.88%	-0.15%	0.31%	-0.15%	1.50%	-0.03%	0.87%	1.23%
gdbj4	7.74%	6.03%	-1.01%	-2.80%	-1.95%	-1.63%	-2.07%	-2.58%	5.87%
gdbj5	9.43%	2.16%	2.97%	2.84%	4.13%	-1.77%	1.43%	0.69%	-1.22%
gdbj6	9.34%	2.82%	6.30%	4.65%	8.88%	0.55%	7.11%	3.89%	1.58%
gdbj7	10.96%	-0.90%	-6.37%	2.22%	3.06%	2.29%	-2.25%	-5.80%	-5.37%
gdbj8	7.91%	-1.94%	-1.01%	-0.28%	-0.28%	0.62%	-0.27%	-0.73%	-1.85%
gdbj9	12.66%	3.04%	0.50%	6.37%	10.18%	-0.44%	-0.60%	0.18%	3.42%
gdbj10	18.94%	-2.34%	1.94%	-0.13%	1.68%	1.60%	-0.54%	-1.48%	-3.02%
gdbj11	9.83%	-1.02%	-0.56%	0.69%	-1.03%	1.02%	-1.79%	-3.30%	0.31%
gdbj12	6.45%	-0.13%	0.02%	-0.37%	-0.17%	0.11%	-0.37%	-0.05%	-0.37%
gdbj14	63.47%	-1.92%	2.60%	0.85%	-12.37%	-1.92%	-17.96%	-13.95%	-8.90%
gdbj15	97.19%	-26.36%	-23.99%	-9.15%	-0.20%	1.92%	-18.06%	-22.43%	-41.20%
gdbj16	51.40%	-6.64%	-8.24%	-22.69%	-0.94%	-12.42%	-10.67%	-10.58%	-18.62%
gdbj17	38.89%	-6.45%	-4.00%	-11.73%	11.59%	16.64%	-3.47%	-9.49%	2.04%
gdbj18	25.51%	7.08%	8.09%	-8.09%	-6.61%	1.33%	1.81%	-0.75%	1.54%
gdbj19	43.35%	53.53%	53.53%	53.53%	53.53%	54.00%	53.53%	53.53%	54.23%
gdbj20	48.57%	-5.44%	-11.78%	-5.80%	-2.37%	-9.37%	-0.10%	-7.88%	-20.67%
gdbj21	37.45%	-4.06%	-11.60%	2.60%	-0.26%	-8.64%	-5.42%	-2.06%	-4.08%
gdbj22	24.37%	3.39%	-7.25%	-6.72%	-7.97%	-2.13%	-9.90%	-11.13%	-1.51%
gdbj23	14.06%	10.07%	2.96%	0.75%	2.72%	-0.63%	-0.57%	2.68%	-6.53%
average		1.58%	0.06%	0.26%	**2.85%**	1.82%	-0.62%	-1.59%	-2.44%

Tableau 4.6 Percentage of improvement of the combinations against the best solution for *gdbj* instances.

Network	Best Improvement	Base	Base, O_4	Base, O_5	Base, O_6	Base, O_7	Base, O_4, O_5	Base, O_4, O_5, O_6	All
mine1 A	4.28%	-1.83%	-1.99%	-1.16%	-1.62%	-1.05%	-1.05%	-1.62%	-0.97%
mine1 B	5.88%	-0.11%	0.12%	-0.20%	0.09%	-0.11%	-0.53%	-0.19%	0.08%
mine2 A	4.37%	0.22%	0.22%	-0.63%	0.86%	0.30%	0.54%	0.35%	-0.38%
mine2 B	6.85%	-0.17%	0.17%	0.44%	0.26%	-0.72%	0.10%	-0.69%	0.37%
mine3 A	2.52%	0.76%	-0.60%	1.31%	0.65%	0.29%	0.28%	0.45%	0.21%
mine3 B	3.29%	-2.76%	-2.17%	-1.86%	-1.86%	-2.02%	-2.82%	-2.63%	-3.29%
mine4 A	3.26%	-3.26%	-3.24%	-3.26%	-2.85%	-3.09%	-3.26%	-3.26%	-3.20%
mine4 B	3.12%	-3.12%	-3.08%	-3.12%	-3.12%	-3.12%	-3.12%	-3.12%	-3.12%
mine5 A	3.92%	0.04%	0.75%	1.74%	0.35%	0.57%	-0.24%	1.54%	1.15%
mine5 B	4.97%	0.16%	0.94%	0.88%	1.13%	-0.34%	1.24%	0.74%	0.29%
Average		-1.01%	-0.89%	**-0.59%**	-0.61%	-0.93%	-0.89%	-0.85%	-0.89%

which is the one used in the previous experiment, whose results are shown in Table 4.4 ; and an unlimited truck capacity, where the truck is able to water all the arcs in one run. A comparison among these scenarios can be seen in Table 4.7. Each scenario has three columns : The first column shows the initial solution. The second column shows the average percentage

of improvement. Finally, the third column shows the average computation time.

It can be seen from the results, that the greater the truck capacity, the better the ini-

Tableau 4.7 Comparison between three different vehicle capacities : Restricted, regular and unlimited.

Network	Restricted capacity			Regular capacity			Unlimited capacity		
	Initial solution	average Improvement	Average running time	Initial solution	Average Improvement	Average running time	Initial solution	Average Improvement	Average running time
mine1 A	1329.2	5.57%	2372.7	1308.5	3.14%	2591.5	1287.6	3.10%	2547.8
mine1 B	1248.3	5.94%	3286.5	1248.3	5.79%	3390.0	1248.3	6.14%	3312.1
mine2 A	3269.7	2.45%	2345.1	3325.7	4.56%	2540.6	3324.4	3.94%	2220.4
mine2 B	3252.2	7.01%	3548.3	3245.7	6.82%	3742.1	3266.3	8.10%	2824.4
mine3 A	5812.8	3.23%	7200.6	5812.0	2.94%	7200.6	5755.1	1.26%	7200.7
mine3 B	5707.1	1.99%	7202.5	5663.0	0.86%	7202.5	5642.7	0.07%	7202.4
mine4 A	4183.2	2.68%	7200.7	4184.2	0.08%	7201.2	4119.5	0.06%	7201.2
mine4 B	4065.5	1.21%	7202.2	4056.8	0.00%	7202.5	3988.6	0.10%	7203.1
mine5 A	2494.7	4.64%	4038.1	2487.8	4.66%	4248.9	2393.5	1.33%	3892.9
mine5 B	2428.6	7.61%	6560.0	2428.6	5.59%	6740.0	2366.4	5.90%	6456.5

tial solution. All instances, except *mine2*, show this behavior. On the other hand, there is no indication that the percentage of improvement of the value of the objective function is influenced by the truck capacity. This suggests that the low percentage of improvement with respect to *gdbj* instances rests in the shape of the networks. Evidence suggests that the initial solution is an accurate solution to the mine-shaped networks. This is also supported by the low percentage of improvement.

4.6 Conclusions and future work

We proposed a mathematical model for the PCARP with inventory constraints specifically applied to watering the roads in open pit mines. The objective is to minimize the formation of airborne dust particles due to the lack of humidity on the roads. This model has been tractable for small networks with as many as five trucks for no more than 30 time periods. We developed an adaptive large neighborhood search algorithm (ALNS) to solve larger networks for a greater number of time periods. The ALNS was able to solve networks with as many as 35 to 55 edges, performing 25000 iterations in 7200 seconds or less.

Eight destroy/repair operators were used to improve the initial solution. When they were tested separately, the use of operator O_3 resulted in a better solution than the other operators in most of the instances tested. Diversification operators, namely O_1, O_2 and O_3 performed better than intensification operators in all instances tested.

To test the ALNS algorithm, different combinations of the destroy/repair operators were used to improve the initial solution. Based on the average value of the objective function for the instances tested, the combinations that resulted in a better improvement of the objective function were the ones containing O_1, O_2, O_3, O_8 and either O_5 or O_6.

For the instances that correspond to the road network of real open-pit mines, the proposed ALNS algorithm results in a small improvement of value of the objective function of the initial solutions, less than 10% and 0% in some cases. We can conclude that the initial solution obtained for these networks results in an accurate solution to the problem. However, further improvement may be achieved by changing the location of the water depot or adding more than one depot. Future work on this problem includes changing it into a Location Routing Problem (LRP) in order to find the location of the depot that maximizes dust particle retention.

4.7 Acknowledgements

This work was partially funded by the National Council for Science and Technology (CONACYT) of the Government of Mexico. The authors would like to thank the referees for all the useful comments that helped improve the quality of this article.

CHAPITRE 5

ARTICLE 3 : LOCATION ARC ROUTING PROBLEM WITH INVENTORY CONSTRAINTS

Cet article a été soumis pour revue dans le journal *Journal of the Operational Research Society* le 11 mars 2014 sous le titre *Location of water depots in open pit-mine networks*. On a reçu les commentaires des arbitres le 12 mai 2014. Les corrections suggérées ont été faites et l'article a été soumis pour revue dans le journal *Computers & Operations Research* le 9 juin 2014 sous le titre *Location arc routing problem with inventory constraints*.

Cet article poursuit l'analyse de l'arrosage des routes dans les mines à ciel ouvert. Un ensemble de camions-citerne arrose de l'eau sur les routes d'une mine à ciel ouvert afin de supprimer des nuages de poussière. Ce qui doit être fait périodiquement parce que l'humidité sur les routes est consommée en fonction du temps. Les camions ont une capacité limitée et doivent retourner à un dépôt principal pour le remplissage. Dans la section précédente, les exemplaires ont été testés pour l'amélioration en utilisant l'algorithme ALNS avec un seul dépôt. Après qu'une amélioration de la valeur de la fonction objectif ait été obtenue, il a été proposé un changement dans le nombre de dépôts le long des réseaux pour réduire les coûts de pénurie et de routage. Deux types expériences ont été faites. La première est un changement de l'emplacement d'un seul dépôt. La seconde est une comparaison entre trois différentes méthodes de localisation pour placer plusieurs dépôts au long du réseau. Dans tous les cas, il y a un dépôt considéré comme dépôt principal qui fonctionne comme le lieu d'où partent et où reviennent les camions au début et à la fin de l'horizon de temps.

Deux types de décisions sont prises dans ce problème, appelé problème de localisation et routage : la décision de la localisation des dépôts et la façon de servir les clients situés sur les arêtes d'un réseau. La majorité des applications des problèmes de localisation et routage apparaissent dans le contexte de tournées sur les nœuds. Quelques exemples, qui seront décrits dans l'article, sont trouvés pour les problèmes de tournées sur les arcs.

Les méthodes de localisation testées suivent une méthode *location, allocation and routing* (L-A-R) où on suggère une localisation initiale de dépôts, on affecte les arêtes du réseau aux camions et, finalement, un trajet est formé pour chaque camion pour servir les arêtes sélectionnées.

La contribution de cet article se résume comme suit :

On propose une méthode de localisation des dépôts qui est appliquée dans le contexte de problèmes de tournées sur les arcs à un problème périodique. Il n'y a pas d'application des méthodes de localisation pour ce type des problèmes.

Location arc routing problem with inventory constraints

Juan Pablo Riquelme Rodríguez, Michel Gamache, André Langevin
CIRRELT, GERAD and Department of Mathematics and Industrial Engineering
École Polytechnique de Montréal

5.1 Abstract

Dust suppression of hauling roads in open pit mines is done by periodically spraying water from a water truck. The objective of this article is to present a method for locating water depots along the road network so that penalty costs for the lack of humidity in roads and routing costs are minimized. Because the demands are located on the arcs of the network and the arcs require service more than once in a time horizon, this problem belongs to the periodic capacitated arc routing domain. We compare three methods for finding the initial depot location using an L-A-R (location, allocation and routing) approach combined with an adaptive large neighborhood search to improve the solution. This method is the first one used for depot location in periodic arc routing problems.

Keywords : Location arc routing problem, Adaptive large-neighborhood search, Periodic capacitated arc routing problem.

5.2 Introduction

5.2.1 Location arc routing problems

Arc routing problems find the routes that satisfy a set of customers located on the arcs of a network while minimizing the associated routing cost. When the objective is to determine the best location of a set of depots so that routing costs are minimized, the problem becomes a location arc routing problem (LARP). The two tasks, locating the depots and finding routes to serve the customers, are performed simultaneously (Levy et Bodin, 1989).
A classification of location routing problems can be found in Lopes *et al.* (2013). Although many examples refer to location problems in the context of node routing problems, i.e., the demands are located on the nodes of the network, some applications can be found in the arc routing domain. A review of the methods used in LARPs can be found in Lopes *et al.* (2014). The first LARP application was presented in 1989 by Levy et Bodin (1989) and involved finding the best locations for postal carriers to park their vehicles in preparation for mail delivery. The authors introduced a location, allocation, and routing (L-A-R) approach in which

the locations for the depots are selected, then the edges to be served are assigned to each depot, and finally routes are built to serve those edges. Ghiani et Laporte (1999) approached the location rural postman problem (LRRP) by transforming it into a rural postman problem (RPP) where there are no bounds on the number of depots and using a branch and cut method to solve it. Ghiani and Laporte Ghiani *et al.* (2001) reviewed the common applications of LARP, such as mail delivery, garbage collection, and street maintenance. The review covered common heuristics used to solve LARP problems including the abovementioned L-A-R and the A-R-L, in which customers are first assigned to a vehicle route, then the route is formed, and finally the depot locations are determined.

Other applications where location decisions are made in the arc routing domain include garbage collection using mobile depots (Del Pia et Filippi, 2006). Small-capacity trucks move along the streets, collecting garbage and delivering their contents to larger trucks used as temporary depots. The authors use a variable neighborhood descent to schedule meetings of the two types of vehicles so that the use of small trucks reduces the number of returns to the main depot. A similar application was presented in Amaya *et al.* (2007), where one type of vehicle is used to paint street lines while a second type is used to refill at specific points in the network.

For this work, we consider an arc routing problem in which a fleet of homogeneous vehicles with limited capacity provide service to the edges of the network. The edges need to be visited more than once in a time horizon. Because several visits are scheduled, the vehicles need to go back to the depot in order to refill and start a new route. This problem is called periodic capacitated arc routing problem (PCARP). The objective is to locate a number of depots along the network so that the refill process can be improved.

5.2.2 Periodic capacitated arc routing problem

The periodic capacitated arc routing problem, or PCARP, was introduced in Lacomme *et al.* (2002) for a garbage collection problem in which the demands on the arcs were different from one period to the next one, and a solution was needed for the whole time horizon instead of for individual periods. This problem was shown to be NP-hard because it contains the capacitated arc routing problem (CARP) as a special case Lacomme *et al.* (2002). CARP was shown to be NP-hard in Golden et Wong (1981). Some applications of the PCARP that have been studied in literature include the already mentioned garbage collection, road monitoring Marzolf *et al.* (2006) and road maintenance and surveillance Monroy *et al.* (2011). Due to the complexity of the problem, heuristic and metaheuristic algorithms have been used to solve large PCARP instances. These algorithms include three heuristic algorithms were

proposed in Chu *et al.* (2005), a memetic algorithm Lacomme *et al.* (2005), a scatter search algorithm Chu *et al.* (2006) and an ant cololy optimization algorithm Kansou et Yassine (2009).

A special case combines the PCARP problem with inventory management (PCARP-IC). The arcs of a network act as customers that require certain quantity of material in stock. The inventory is replenished periodically by means of delivery vehicles whit limited capacity. Vehicles return to a depot for refill and start a new delivery route. Applications include the dust suppression in open-pit mine roads Li *et al.* (2008)Riquelme Rodríguez *et al.* (2013) and plants watering in street medians and sidewalks. This problem was introduced as such in Riquelme Rodríguez *et al.* (2013). The authors propose a mathematical model which is capable of solving small instances. An adaptive large neighborhood search was proposed to solve larger instances of this problem in Riquelme Rodríguez *et al.* (2014). In order to accelerate the refill of vehicles, depots are strategically located along the network. To the best of our knowledge, location decisions have not been studied for a periodic capacitated arc routing problem. The only studies combining location and routing decisions for periodic applications are presented in Prodhon (2011) and Prodhon et Prins (2008). Both refer to node routing problems. The contribution of this article is a mathematical model as well as a heuristic algorithm to deal with a location problem in the periodic arc routing domain.

The article is organized as follows. In Section 5.3, we present the definition of the problem of road watering in open-pit mines and the mathematical model. Our algorithm is presented in Section 5.4, and the results are discussed in Section 5.5. Finally, Section 5.6 presents concluding remarks.

5.3 Mathematical model

5.3.1 Problem definition

In open-pit mines, when vehicles travel along the roads, dust clouds are formed. The most cost-effective method for suppressing dust in these temporary roads is to periodically spray water over them (Neulicht et Shular, 1998). Due to evaporation and traffic volume, humidity is lost and needs to be replenished. The roads in the network can be traversed in any direction. They are classified according to their priority, where the roads with higher traffic volume have a higher priority. A set of edges E represents the roads. A penalty cost is assigned for having a lower level of humidity than the required one to ensure dust particle retention. Figure 5.1 shows the humidity level, H_{ij}^t, of edge $[i,j] \in E$. $\overline{h_{ij}}$ is the required level of humidity to ensure dust retention for edge $[i,j] \in E$. After a quantity of water, q_{ij}^{kt}, is delivered by vehicle k at time t, humidity is consumed until the next service. Figure 5.1 a. shows the shortage

of humidity that occurs when H_{ij}^t is less than $\overline{h_{ij}}$. Figure 5.1b. shows the same situation for a discretized time horizon divided in time periods of equal duration. H_{ij}^t is considered to be constant during a time period.

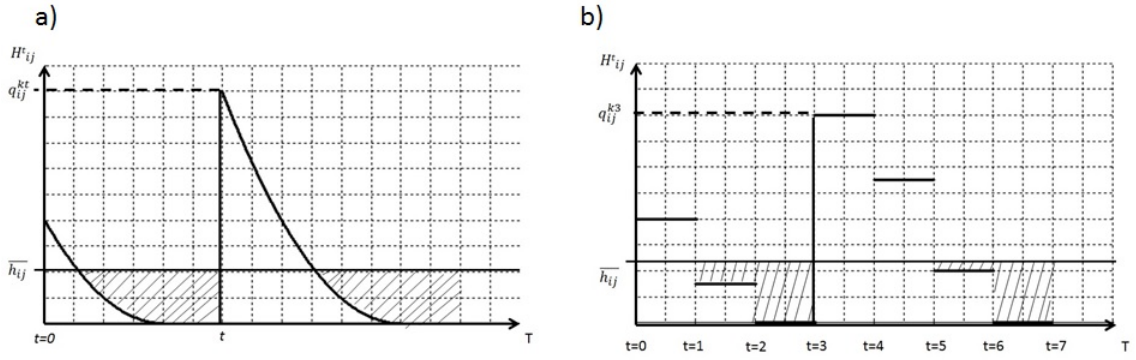

Figure 5.1 Humidity level of edge $[i, j]$

Because water trucks have a limited capacity, they reload at the depot before starting a new route. The objective is to find the location of water depots along the mine road network so that the penalty cost for lack of humidity and the routing cost are minimized. For this problem, we consider that the depots can handle any number of vehicles.

This problem combines strategic and operational decisions. The placement of depots is a long term decision, therefore, the performance of the vehicles is tested on different scenarios. A scenario is created when the values of some parameters are changed. For example, at the beginning of the time horizon the roads may have different levels of humidity. Each initial humidity level represents a scenario.

5.3.2 Mathematical model

The model presented in this section is based on the model presented in Riquelme Rodríguez *et al.* (2014) that aims to minimize operational costs such as penalty and routing costs when one depot and a fleet of homogeneous vehicles are considered. We include both of these costs tested under different scenarios in order to minimize long term costs such as vehicle and depot placement.

Consider a time horizon that corresopnds to one working shift divided in T time periods. A time period is the amount of time it takes a water truck to cover a constant distance D at a constant speed. For example, a truck traveling at $20km/h$ can cover a distance $D = 300m$ in approximately 1 minute (54 seconds). We assume service and deadheading speeds for the water truck are the same. Even when the truck has a faster speed during deadheading,

there are several factors affecting this speed such as the presence of other trucks in the same road (Li *et al.*, 2008) and the condition of the road (Thompson et Visser, 2000). The analisys of different truck speed for the model with one depot and one vehicle is presented in Riquelme Rodríguez *et al.* (2013). Consider a mixed network $G(N, E \cup A)$, where N is the set of nodes, and E is the set of edges that correspond to the roads of the mine network. A is the set of arcs that indicate the direction of the traversal of each edge. $B_1 \subseteq A$ is the set of arcs such that for each edge $[i, j] \in E$ there are two artificial arcs $(i, j), (j, i) \in B_1$. $B_2 \subseteq A$ is the set of artificial loops located at each node $i \in N$ where it is possible to place a depot. A loop $(i, i) \in B_2$ is used to simulate the refill of a vehicle. Note that $B_1 \cap B_2 = \emptyset$.
Consider the following parameters of the model :

- C is the number of scenarios that simulate different conditions.
- K is the number of trucks of with homogeneous capacity Q^{max}.
- T is the number of time periods in the time horizon.
- A penalty cost P_{ij} is given to edge $[i, j] \in E$ if the humidity level is below the minimum humidity level. $\overline{h_{ij}}$, required to ensure particle retention.
- hc_{ij} is the cost of watering arc $(i, j) \in B_1$ or refill at loop $(i, i) \in B_2$.
- tr_{ij} are the cost traversing arc $(i, j) \in B_1$ without service or waiting at $(i, i) \in B_2$.
- cv is the cost of purchasing a vehicle and cd_b is the cost of establishing a depot in location $b \in B_2$.
- g_{ij} is the percentage of humidity loss due to the traffic volume in edge $[i, j] \in E$.
- e^{tc} is the evaporation factor due to the time period $t \in \{0, \ldots, T\}$ and scenario $c \in \{1, \ldots, C\}$.
- I^c_{ij} represents the initial humidity level of edge $[i, j] \in E$ in scenario c.
- H^{max}_{ij} is the maximum humidity level allowed in edge $[i, j]$.
- $MinDep$ and $MaxDep$ are, respectively, the minimum and maximum number of depots allowed in the network. In an open-pit mine, the number of potential location sites is limited by the fact that the mine's topologies are in perpetual evolution.
- d_{ij} is the number of time periods required to travel along arc $(i, j) \in A$.

- $\Omega \subseteq A$ is the set of arcs that require more than one time period to be traversed, i.e., $d_{ij} > 1$.

Consider the following variables of the model :

- H^{tc}_{ij} is the humidity level of edge $[i, j] \in E$ at time t of scenario c.
- $Y^{ktc}_{ij} = 1$ if vehicle $k \in \{1, \ldots, K\}$ traverses arc $(i, j) \in B_1$ without service or waits at $(i, i) \in B_2$ at the beginning of time t of scenario c, 0 otherwise.
- $X^{ktc}_{ij} = 1$ if vehicle k waters arc $(i, j) \in B_1$ or refills at $(i, i) \in B_2$ at the beginning of

time t of scenario c, 0 otherwise.

- $Z_b = 1$ if arc $b \in B_2$ is chosen as a location for a depot, 0 otherwise.
- $R_k = 1$ if vehicle k is used, 0 otherwise.
- Q^{ktc} is the quantity of water in vehicle k at the beginning of time t of scenario c.
- q_{ij}^{ktc} is the quantity of water delivered to edge $[i,j] \in E$ by truck k at the beginning of time t of scenario c.
- q_b^{ktc} is the quantity of water to be refilled to vehicle k at depot $b \in B_2$, at the beginning of time t of scenario c.
- $S_{uv}^{ktc} = 1$ if vehicle k is between nodes u and v of arc $(u,v) \in \Omega$ at the beginning of time t of scenario c, 0 otherwise.
- $w_{ij}^{tc} = \max\{0, \overline{h_{ij}} - H_{ij}^{tc}\}$, $\forall [i,j] \in E$. It is the difference in humidity levels when $\overline{h_{ij}} \geq H_{ij}^{tc}$ as depticted in Figure 5.1, 0 otherwise.

The complete model is as follows :

$$\min \frac{1}{C}\left[\sum_{c=1}^{C}\sum_{[i,j]\in E}\sum_{t=0}^{T} P_{ij} w_{ij}^{tc} + \sum_{c=1}^{C}\sum_{(i,j)\in A}\sum_{k=1}^{K}\sum_{t=0}^{T}(hc_{ij} X_{ij}^{ktc} + tr_{ij} Y_{ij}^{ktc})\right] + \sum_{k=1}^{K} cv R_k + \sum_{b\in B_2} cd_b Z_b \tag{5.1}$$

subject to :

$$w_{ij}^{tc} \geq \overline{h_{ij}} - H_{ij}^{tc} \qquad \forall [i,j] \in E,\ t \in \{0,\ldots,T\},\ c \in \{1,\ldots,C\} \quad (5.2)$$

$$H_{ij}^{(t+1)c} = (1 - (e^{tc} g_{ij}))H_{ij}^{tc} + \sum_{k=1}^{K} q_{ij}^{k(t+1)c} \qquad \forall [i,j] \in E,\ t \in \{0,\ldots,T-1\},\ c \in \{1,\ldots,C\} \quad (5.3)$$

$$H_{ij}^{0c} = I_{ij}^{c} \qquad \forall [i,j] \in E,\ c \in \{1,\ldots,C\} \quad (5.4)$$

$$q_{ij}^{ktc} \leq H_{ij}^{max}(X_{uv}^{ktc} + X_{vu}^{ktc}) \qquad \forall \{[i,j] \in E | i = u, j = v; (u,v), (v,u) \in A\},\ t \in \{0,\ldots,T\},\ k \in \{1,\ldots,K\},\ c \in \{1,\ldots,C\} \quad (5.5)$$

$$H_{ij}^{tc} \leq H_{ij}^{max} \qquad \forall [i,j] \in E,\ t \in \{0,\ldots,T\},\ c \in \{1,\ldots,C\} \quad (5.6)$$

$$\sum_{[i,j] \in E} q_{ij}^{ktc} \leq Q^{ktc} \qquad \forall t \in \{0,\ldots,T\},\ k \in \{1,\ldots,K\},\ c \in \{1,\ldots,C\} \quad (5.7)$$

$$Q^{k0c} = Q^{max} \qquad \forall\, k \in \{1,\ldots,K\},\ c \in \{1,\ldots,C\} \quad (5.8)$$

$$Q^{ktc} \leq Q^{max} \qquad \forall t \in \{0,\ldots,T\},\ k \in \{1,\ldots,K\},\ c \in \{1,\ldots,C\} \quad (5.9)$$

$$Q^{k(t+1)c} = Q^{ktc} - \sum_{[i,j] \in E} q_{ij}^{ktc} - \sum_{b \in B} q_{b}^{ktc} \qquad \forall t \in \{0,\ldots,T-1\},\ k \in \{1,\ldots,K\},\ c \in \{1,\ldots,C\} \quad (5.10)$$

$$q_{b}^{ktc} = -Q^{max} X_{b}^{ktc} \qquad \forall t \in \{0,\ldots,T\},\ k \in \{1,\ldots,K\},\ b \in B_2,\ c \in \{1,\ldots,C\} \quad (5.11)$$

$$\sum_{c=1}^{C}\sum_{(i,j)\in A}\sum_{T=0}^{T} X_{ij}^{ktc} \leq R_k TC \qquad \forall k \in \{1,\ldots,K\} \tag{5.12}$$

$$\sum_{c=1}^{C}\sum_{t=0}^{T}\sum_{k=1}^{K}(X_b^{ktc} + Y_b^{ktc}) \leq Z_b KTC \qquad \forall b \in B_2 \tag{5.13}$$

$$MinDep \leq \sum_{b\in B_2} Z_b \leq MaxDep \tag{5.14}$$

$$Z_{dep} = 1 \tag{5.15}$$

$$X_{ij}^{ktc} + Y_{ij}^{ktc} \leq 1 \qquad \forall (i,j) \in A,\ t \in \{0,\ldots,T\},\ k \in \{1,\ldots,K\},\ c \in \{1,\ldots,C\} \tag{5.16}$$

$$\sum_{k=1}^{K} X_{ij}^{ktc} \leq 1 \qquad \forall (i,j) \in A,\ t \in \{0,\ldots,T\},\ c \in \{1,\ldots,C\} \tag{5.17}$$

$$\sum_{(i,j)\in A|i=dep} X_{ij}^{k0c} + Y_{ijc}^{k0c} = 1 \qquad \forall k \in \{1,\ldots,K\},\ c \in \{1,\ldots,C\} \tag{5.18}$$

$$\sum_{(i,j)\in A|j=dep} X_{ij}^{kTc} + Y_{ij}^{kTc} = 1 \qquad \forall k \in \{1,\ldots,K\},\ c \in \{1,\ldots,C\} \tag{5.19}$$

$$X_{ij}^{ktc} + Y_{ij}^{ktc} \leq \sum_{l|(j,l)\in A} X_{jl}^{k(t+d_{ij})c} + Y_{jl}^{k(t+d_{ij})c} \qquad \forall (i,j) \in A,\ t \in \{0,\ldots,T\},\ k \in \{1,\ldots,K\},\ c \in \{1,\ldots,C\} \tag{5.20}$$

$$X_{ij}^{ktc} + Y_{ij}^{ktc} \leq S_{ij}^{k(t+m)c} \qquad \forall (i,j) \in \Omega,\ m \in \{0,\ldots,d_{ij}-1\},\ t \in \{0,\ldots,T-m\},\ k \in \{1,\ldots,K\},\ c \in \{1,\ldots,C\} \tag{5.21}$$

$$\sum_{(i,j)\in A\setminus\{(u,v)\}} (X_{ij}^{ktc} + Y_{ij}^{ktc}) \leq 1 - S_{uv}^{ktc} \qquad \forall t \in \{0,\ldots,T\},\ (u,v) \in \Omega,\ k \in \{1,\ldots,K\},\ c \in \{1,\ldots,C\} \tag{5.22}$$

$$H_{ij}^{tc},\ w_{ij}^{tc} \geq 0 \qquad \forall [i,j] \in E, t \in \{0,\ldots,T\},\ c \in \{1,\ldots,C\} \tag{5.23}$$

$$q_{ij}^{ktc} \geq 0 \qquad \forall [i,j] \in E, t \in \{0,\ldots,T\},\ k \in \{1,\ldots,K\},\ c \in \{1,\ldots,C\} \tag{5.24}$$

$$q_{b}^{ktc} \leq 0 \qquad \forall t \in \{0,\ldots,T\},\ k \in \{1,\ldots,K\},\ b \in B_2,\ c \in \{1,\ldots,C\} \tag{5.25}$$

$$X_{ij}^{ktc},\ Y_{ij}^{ktc} \in \{0,1\} \qquad \forall (i,j) \in A, t \in \{0,\ldots,T\},\ k \in \{1,\ldots,K\},\ c \in \{1,\ldots,C\} \tag{5.26}$$

$$Q^{tc} \geq 0 \qquad \forall t \in \{0,\ldots,T\},\ c \in \{1,\ldots,C\} \tag{5.27}$$

$$S_{uv}^{ktc} \in \{0,1\} \qquad \forall (u,v) \in \Omega, t \in \{0,\ldots,T\},\ c \in \{1,\ldots,C\},\ k \in \{1,\ldots,K\} \tag{5.28}$$

$$Z_b \in \{0,1\} \qquad \forall b \in B_2 \tag{5.29}$$

$$R_k \in \{0,1\} \qquad \forall k \in \{1,\ldots,K\} \tag{5.30}$$

The objective function (5.1) minimizes the total cost. The penalty cost for a shortage of humidity and the routing cost of watering and deadheading are the result of the average cost of all scenarios. It is added to the vehicle cost and the depot location cost.
Constraints (5.2) define variable w_{ij}^{tc} as the difference between the required and actual humidity levels at a time t for each scenario. If positive, this difference is penalized in the objective function. Constraints (5.3) estblish the humidity level for the next time period, for each scenario. Constraints (5.4) establish the initial humidity level of each edge for each scenario. Constraints (5.5) and (5.6) limit, respectively, the amount of water delivered and the humidity level to the maximum humidity level. Constraints (5.7) are vehicle capacity constraints. Constraints (5.8) determine the initial quantity of water available in each vehicle. Constraints (5.9) limit the vehicle capacity at each time period to the maximum capacity. Constraints (5.10) determines the quantity of water in the vehicle at each time period. Constraints (5.11) allow the truck refilling at each depot. Constraints (5.12) allow the use of trucks. Constraints (5.13) allow the usage of a selected depot. Constraint (5.14) establishes the limits on the number of depots. Equation (5.15) determines the main depot, $dep \in N$. This is the depot from which vehicles depart at the beginning of the time horizon using constraints (5.18) and to which they return at the end of it according to constraints (5.19). Constraints (5.16) limit the vehicle to either water or traverse an edge at a specific time period. Constraints (5.17) state that an edge can be serviced by only one vehicle at the same time period. Constraints (5.20) are flow conservation constraints for all arcs in the network. Constraints (5.21) and (5.22) ensure that the vehicle stays on the same edge during the number of time periods it takes to service or traverse it. Finally, constraints (5.23) to (5.30) define the variables of the model.

We added the following constraints to break the symetry in the solution tree :

$$Y_{00}^{ktc} = 1 \qquad \forall k \in \{1, \ldots, K\},\ t \in \{0, \ldots, k-1\},\ c \in \{1, \ldots, C\} \tag{5.31}$$

$$\sum_{(0,j)\in A} X_{0j}^{ktc} + Y_{0j}^{ktc} = 1 \qquad \forall k \in \{1, \ldots, K\},\ t = k,\ c \in \{1, \ldots, C\} \tag{5.32}$$

Restrictions (5.31) ensures that vehicles stay at the initial depot for an increasing number of periods. The first vehicle will stay at the depot until the first time period, the second vehicule will stay until the second period and so on. Equations (5.32) ensure vehicles leave the depot one at each period.
This model was coded in cplex OPL and solved for a network of 8 nodes and 11 edges, $K = 5$ vehicles and 10 scenarios. The scenarios included 10 different values for I_{ij}^{c}. The stopping criterion was 2 hours or reaching 2% relative gap. The program was able to solve this instance for less than 18 time periods before meeting the stopping criterion. These parameters resluted in a problem with 160000 restrictions and 97000 variables. For larger instances and a larger time horizon, a heuristic method is required.

5.4 Location and routing algorithms

Two LARP heuristics were described in Ghiani *et al.* (2001) : The Location-Allocation-Routing (L-A-R) heuristic and the Allocation-Routing-Location (A-R-L). In the first approach, the depots are located, the edges of the network are assigned to the depots and finally a route is formed. In the second one, the edges are allocated and a route is formed for each vehicle, the depots are located in the end. For our problem, the location of depots affects the resulting route. Vehicles refill in an existing depot in order to start a new service route, thus the beginning and end of a route is determined by the location of the nearest depot. We use the L-A-R approach with the difference that in the allocation phase, the edges of the network are assigned to vehicles instead of assigned to depots. We also add an improvement phase at the end of the last step by using an adaptive large neighborhood search heuristic that changes the initial allocation and routing by means of a series of destroy - repair operators. Both the L-A-R approach and the improvement phase are explained in this section.

5.4.1 Location algorithms

The first step of the L-A-R approach is the location of depots in the nodes of the network. We use three methods for the initial location of depots.

1. Levy and Bodin location method.

This method was proposed by Levy and Bodin Levy et Bodin (1989) for a mail delivery application. In this problem, postal carriers park their vehicles at specific locations and walk to deliver mail. The objective is to find the best location to park the vehicles. The authors follow the L-A-R approach. The location phase is done by classifying the nodes of the network in decreasing order of their attractiveness measure and selecting the nodes using a separation criterion.

We adapt location phase of the Levy and Bodin's L-A-R method to our problem as follows :

- Determine the nodes that can be used as depots. For this problem we are assuming every node is a potential location for a depot.
- Arrange the nodes in decreasing order of their *attractiveness measure* (AM). A node has a higher AM if there are more incident nodes to it and the priority of the incident nodes is higher.

$$\text{AM} = A_1\text{*Number of incident nodes} + A_2\text{*Sum of the priorities of incident nodes}$$

where A_1 and A_2 are the weights given to each contributing factor of the attractiveness measure.

- Choose the nodes from the list using a separation criterion. The next node must have a minimum distance from the last selected node, otherwise, it is discarded and the next node is tested. The distance between nodes is calculated using Dijkstra's algorithm. The selected node is removed from the AM list and added to the depot list. The minimum distance is a percentage of the total distance the truck is able to water having full capacity.
- The process is repeated until there are no nodes left in the AM list.

2. Random location.

This method assigns P nodes randomly as depots, where P is the number of depots obtained by Levy and Bodin's method. Assuming the more depots there are, the higher the cost, a comparison can only be made if all the methods have the same number of depots. This method is used as base line for comparison of the other two.

3. Minimum distance.

This method distributes depots along the nodes of the network so that the distance from the edges to the selected depots is minimized. Consider $X_{ij} = 1$ if edge j is assigned to depot i, 0 otherwise ; and $Y_i = 1$ if a depot is placed at node i, 0 otherwise. C_{ij} represents the shotest path from depot i to edge j, P is the number of depots obtained using Levy and Bodin's method and E is the number of edges.

$$\min \sum_i \sum_j C_{ij} X_{ij} \tag{5.33}$$

subject to :

$$\sum_i Y_i = P \tag{5.34}$$

$$\sum_i X_{ij} = 1 \qquad \forall j \tag{5.35}$$

$$\sum_j X_{ij} \leq E Y_i \qquad \forall i \tag{5.36}$$

$$X_{ij} \in \{0, 1\} \qquad \forall i, j \tag{5.37}$$

$$Y_i \in \{0, 1\} \qquad \forall i \tag{5.38}$$

The objective function (5.33) minimizes the shortest distance between the selected depots and the edges of the network. Equation (5.34) establishes the number of depots. Equations (5.35) ensures all edges are assigned to a depot. Constraints (5.36) allows an edge to be assigned to node i if it was selected as depot. Constraints (5.37) and (5.38) define the variables.
The three methods result in a list of nodes that are used to hold a depot. The first node of the list is considered the *main* depot. The main depot is the place where vehicles start and where they return at the beginning and end of the time horizon. In the case of the Levy and Bodin method it is the node with the highest attractiveness measure.
The next two steps, allocation and routing, are based on the initial solution of the algorithm proposed by Riquelme Rodríguez *et al.* (2014) for the PCARP with inventory constraints.

5.4.2 Allocation

The network is partitioned in K sets of edges, where K is the number of vehicles. This procedure was described in Riquelme Rodríguez *et al.* (2014) and is based on the *cluster first-route second* algorithm used in Monroy *et al.* (2011) for the PCARP with irregular services. It can be summarized as follows :

1. Select K edges called *seeds* as far away from each other. The shortest path between edges e_1 and e_2, $SP_{e_1 e_2}$, is calculated using Dijkstra's algorithm. The first seed s_1, is the one with the highest $SP_{0 s_1}$, where 0 represents the main depot selected by any of the location methods. From a set of selected seeds $\{s_1, \ldots, s_h\}$, when $h < k$, select edge e as s_{h+1} so that $SP_{0e}(\prod_{i=1}^{h} SP_{s_i e})$ is maximum.

2. Assign the rest of the edges of the network by minimizing the sum of the lenghts of shortest paths from the edges to the seeds selected in step 1.

5.4.3 Routing

A route is calculated for each one of the vehicles by means of a constructive algorithm, first proposed in Riquelme Rodríguez *et al.* (2014). Each vehicle can serve only the edges assigned to it in the allocation phase, but it can traverse any edge. A route with a starting and ending depot is called a *run*. The main depot represents the start of the first run and the end of the last one. Any vehicle can refill at any depot, including the main depot, before starting a new run. For each run, the total quantity of water used is calculated as well as the time needed to traverse the arcs on that run. Our routing procedure can be sumarized for each one of the vehicles as follows :

1. Start with a list L_1 of available edges, i.e., edges whose humidity level is below the required level and need service. After an edge is serviced in the first run, it will not be available for service (removed from L_1) for subsequent runs until H_{ij}^{kt} is reduced enough so that it is available again.
2. From L_1, select the edge, $[i, j]$, with the highest priority and add it to a list of selected edges. Denote L_2 the list of edges from L_1 that are selected to be serviced.
3. Assign a quantity of water to be delivered equal to $\alpha D d_{ij}$, where α is the water rate in number of liters per meter and D is the distance, in meters, a truck travels in one time period at a constant speed.
4. Order the available neighbors, i.e., adjacent edges to the edges in L_2 that need service, in increasing order of the length of their shortest paths to the departing depot and select the first edge to add to L_2. If no neighbor is available, select the closest available edge to the edges in L_2.
5. Repeat step 4 until the residual amount of water in the truck is less than $\alpha D d_{ij}, \forall [i, j] \notin L_1$ or $L_1 = \emptyset$.
6. Order the edges in L_2 in increasing order of their shortest paths to the departing depot.
7. Connect the first edge to the departing depot calculating the shortest path between them. Connect the rest of the edges in L_2 using the shortest paths between them. The direction in which the edges are traversed is important. An edge $[i, j]$ that is traversed either in the direction of the arc (i, j) or (j, i) is removed from L_2. The process is repeated until $L_2 = \emptyset$.
8. Find the closest depot to the last arc (i, j) in the sequence and find the shortest path to connect them. The set of arcs starting and ending in a depot is called a *run*.

9. Create as many runs as possible within the time horizon, repeating steps 1-8. If the time to complete a run ends after the end of the time horizon, reorganize the path to end the run at the main depot at the end of the time horizon.

5.4.4 Adaptive large-neighborhood search

We use an adaptive large neighborhood search heuristic to improve the solution found by the L-A-R method. ALNS was introduced in Ropke et Pisinger (2006). The ALNS has been used in the arc routing domain for the synchronized arc routing problem Salazar-Aguilar *et al.* (2012) in which snow plowing operations are performed by coordinating a set of vehicles, and for a road marking application Salazar-Aguilar *et al.* (2013). Applications that require the combination of routing and inventory decisions in the node routing domain that use the ALNS can be found in Coelho *et al.* (2012a) and Coelho *et al.* (2012b). For these examples, the customers are located in the nodes of the network instead of the arcs.

The ALNS is an iterative process in which a set of destroy/repair operators, $\{O_1, \ldots, O_n\}$, modify the initial solution. The process is divided in segments. A segment consists of 250 iterations. In each iteration, an operator O_i is chosen using a *roulette-wheel* mechanism, i.e. it is chosen with probability $\rho_i / \sum_{i=1}^{n} \rho_i$, where ρ_i is the weight of O_i. It is defined as $\rho_i = C_i/U_i$ where C_i is the score of O_i and U_i is the number of times it was used during one segment. When a segment starts, C_i is set to 0 for all operators and the weights are updated so the probability of selecting each operator changes according to past performances. The operators with better performances have a higher probability of being selected. At the beginning of the procedure, the weights are the same, $\rho_i = 1/n$. After segment j, the weights for segment $j+1$ are calculated using $\rho_{i,j+1} = \rho_{ij}(1-r) + r\frac{C_i}{U_i}$, where $r \in [0,1]$ is called *reaction factor*. C_i is determined by three scores : σ_1 is awarded if the use of O_i results in a better solution overall ; σ_2 is awarded if the use of O_i results in a better solution than the incumbent one, but not better than the best solution and it has not been explored before ; and σ_3 is awarded if O_i results in a bad solution, accepted with probability $e^{-(f'-f)/\tau}$, being f and f', the current and new solution, while the temperature factor τ, starts with τ_0 and decreases with each iteration $\tau = \tau \times c$, where $0 < c < 1$. τ_0 is calculated from the initial solution so that a solution that is μ% worse than the current solution is accepted with 50% probability Ropke et Pisinger (2006).

We use seven destroy/repair operators from Riquelme Rodríguez *et al.* (2014). The first three operators, O_1, O_2, and O_3, have the purpose of diversification. They randomly exchange the edges that were previoulsy assigned to be served by a specific vehicle in the allocation phase. The rest of the operators are intensification operators. They perform changes in the sequences created after the routing phase. They are summarized as follows :

- O_1 exchanges one randomly selected edge from two lists obtained after the allocation phase.
- O_2 exchanges two adjacent, randomly selected edges from two lists obtained after the allocation phase.
- O_3 exchanges a random number of edges from two two lists obtained after the allocation phase.
- O_4 The service status of a random number of edges is changed from *serviced* to *not serviced* and vice versa.
- O_5 randomly exchanges two sequences of arcs (serviced or not) created after the routing phase. Both sequences are performed by the same vehicle. Because the inserted sequences may not be adjacent to the existing arcs, the shortest path between them is calculated.
- O_6 is similar to O_5 except that the two sequences exchanged are performed by two different vehicles. If total water used from the resulting inserted run in O_5 or O_6 excedes the capacity of the vehicle, thus becoming infeasible, edges are removed starting from the last one until the solution regains feasibility.
- O_7 randomly changes the amount of water to be delivered.

All operators destroy the existing solution and repair it in the same iteration. All repaired solutions are feasible. If an operator results in an unfeasible solution, it is discarded.
Because of the inventory constraints in the mathematical model, i.e., constraints (5.3) and (5.10), that show the level of humidity and vehicle capacity in period $t+1$ from the existing levels in period t, every time an operator performs any change in the solution, the rest of the solution needs to be re-calculated for the rest of the time horizon. For example, if an edge previously not served is served due to operator O_4, it will not be available for the next run, thus the list of edges that can be serviced in the second run is different.

5.5 Test results

5.5.1 Parameters

We performed the tests on the *mine* instances from Riquelme Rodríguez *et al.* (2014). The road networks are taken from 5 real mines, and the number of vehicles used is either 3 or 5. The location parameters used to compute the attractiveness measure, A_1 and A_2, are set to 0.5 and 0.5. The distance percentage D_1 is set to 0.6. This is the highest percentage for which the smallest network *mine1* resulted in more than one depot.
The ALNS parameters are taken from Riquelme Rodríguez *et al.* (2014) in order to compare the results. They were tuned by changing the value of one of them and choosing the one

that delivered the best objective value. Keeping the first parameter fixed, we tested the second for several values. This process was repeated for all the parameters, resulting in $(r, \mu, \sigma_1, \sigma_2, \sigma_3) = (0.1, 0.8, 25, 10, 5)$.

The network parameters are taken from Riquelme Rodríguez *et al.* (2014). The time horizon is $T = 300$ time periods. The traversing cost is the same as the edge distance, i.e., $r_{ij} = d_{ij}$, while the watering cost is set to $c_{ij} = r_{ij} + 2$. To give priority to the penalty costs we set $P_{ij} = \max c_{ij} * priority_{ij}$, where the priority of edge $[i, j]$ is an integer from 1 to 10. The required level of humidity is set to $\overline{h_{ij}} = \alpha D d_{ij}$. I_{ij}^{tc} and H_{ij}^{max} are set to $0.1\overline{h_{ij}}$ and $1.1\overline{h_{ij}}$ respectively ; b_{ij} is set to 3% for low-priority edges and 7% for high-priority edges. $e_t = e_{t-1} + 0.1$ if $\lceil 0.55T \rceil \leq t < \lceil 0.75T \rceil$; $e_t = e_{t-1} - 0.1$ if $\lceil 0.75T \rceil \leq t < \lceil 0.95T \rceil$; and $e_t = 1$ otherwise. These parameters are used to approximate the daytime evaporation rate, which is increased in the second half of the time horizon, as shown in the results for hourly evaporation given in Molina-Martínez *et al.* (2006).

Table 5.1 gives the characteristics of each network ; $|N|$ represents the number of nodes and $|E|$ the number of edges. The last two columns show the number of depots obtained using Levy and Bodin's method when there is no restriction on the main depot (column 5), and when the main depot is located at node 0 (column 6).

All the algorithms were coded in Python and executed on a 1.8 GHz Intel Core i5-3337U notebook PC.

Tableau 5.1 General information on the *mine* instances tested.

Network	Number of vehicles	$\|N\|$	$\|E\|$	Number of depots $dep = n$	Number of depots $dep = 0$
mine1 A	3	21	22	4	5
mine1 B	5	21	22	4	5
mine2 A	3	22	27	7	7
mine2 B	5	22	27	7	7
mine3 A	3	49	53	10	9
mine3 B	5	49	53	10	9
mine4 A	3	51	60	5	6
mine4 B	5	51	60	5	6
mine5 A	3	30	35	5	6
mine5 B	5	30	35	5	6

5.5.2 Alternative location of one depot

In the *mine* instances from Riquelme Rodríguez *et al.* (2014), the depot is located by default at node 0, which is the node located in the farthest node from the pit of the mine. We tested a better location for a 1-depot network placed on the node with the highest degree of attractiveness according to Levy and Bodin's method, hereafter referred to as *node n*. Table 5.2 shows the results. The third column shows the total cost of the initial solution (no ALNS improvement) and is compared with the L-A-R method from Levy and Bodin (column 4), without any ALNS improvement. The best cost between the two is highlighted. Columns 5 and 6 show the percentage of improvement of the initial solution obtained after performing the ALNS algorithm The best result is highlighted. The stopping criterion being 25000 iterations or 7200 seconds.

Tableau 5.2 Value of the objective function with one depot located at node 0 and at node n.

		Initial solution		ALNS	
Network	Number of vehicles	1 depot at node 0	1 depot at node n	1 depot at node 0	1 depot at node n
mine1 A	3	1308.55	**1257.80**	5.48%	**7.84%**
mine1 B	5	1248.35	**1159.82**	6.10%	**13.18%**
mine2 A	3	3325.69	**3317.48**	5.18%	**6.33%**
mine2 B	5	**3245.71**	3274.68	6.88%	**9.43%**
mine3 A	3	5812.03	**5795.60**	1.94%	**1.98%**
mine3 B	5	**5663.04**	5741.13	0.03%	**2.38%**
mine4 A	3	4184.20	**4126.75**	**2.31%**	1.72%
mine4 B	5	4056.81	**3933.53**	**0.97%**	0.00%
mine5 A	3	2487.81	**2448.89**	3.87%	**10.78%**
mine5 B	5	2428.56	**2266.13**	4.88%	**10.39%**

In 8 out of 10 instances of the solutions with no ALNS improvement, the cost is lower when the depot is located in a different node than node 0. Only two instances showed better results when the depot is located at node 0. 8 out of 10 instances showed a better improvement of the initial solution when the depot is located at a node other than 0.

5.5.3 Comparison of one and several depots

The second test is a direct comparison between having one depot and several. We used Levy and Bodin's method to find the depots location. Table 5.3 shows the results for the initial solution (no ALNS improvement). Column 3 shows the cost when there is only one depot at node 0 ($dep = 0$). Column 4 shows the cost when there are P depots (P varies according to the network), but the main depot is located at node 0. Column 5 shows the cost using P depots, being the main depot, the one located at the node with the highest degree of attractiveness according to Levy and Bodin's method ($dep = n$). Table 5.4 shows the same comparison between 1 and several depots after performing the ALNS improvement. The percentages of improvement of the initial solution are shown.

Tableau 5.3 Comparison of the objective function values for one and N depots.

Network	Number of vehicles	1 depot at node 0	N depots $dep = 0$ L-A-R	N depots $dep = n$ L-A-R
mine1 A	3	1308.55	1292.15	**1248.95**
mine1 B	5	1248.35	1176.83	**1170.25**
mine2 A	3	3325.69	3327.21	**3323.06**
mine2 B	5	3245.71	**3115.66**	3157.31
mine3 A	3	5812.03	5752.06	**5739.53**
mine3 B	5	5663.04	5639.34	**5624.56**
mine4 A	3	4184.20	4104.56	**4086.48**
mine4 B	5	4056.81	3972.98	**3918.58**
mine5 A	3	2487.81	2478.83	**2407.66**
mine5 B	5	2428.56	2309.24	**2251.46**

In all the instances, the cost is lower when more depots are included in the network. We note that for one instance, *mine2B*, the solution is better when the main depot is forced to be at node 0.

Based on the ALNS results shown in Table 5.4, the best location for the main depot for network *mine2* is node 0. For the rest of the networks, except *mine3B*, the best location for the main depot is node n.

Tableau 5.4 Comparison of the objective function values for one and N depots after ALNS.

Network	Number of vehicles	1 depot at node 0	N depots $dep = 0$ L-A-R and ALNS	N depots $dep = n$ L-A-R and ALNS
mine1 A	3	1236.86	1082.80	**1082.29**
mine1 B	5	1172.15	858.28	**855.01**
mine2 A	3	3153.40	**2825.41**	2979.19
mine2 B	5	3022.37	**2539.98**	2781.32
mine3 A	3	5699.44	5624.09	**5553.20**
mine3 B	5	5661.11	**5515.76**	5530.12
mine4 A	3	4087.70	4044.91	**4009.93**
mine4 B	5	4017.61	3944.78	**3896.19**
mine5 A	3	2391.64	2285.00	**2217.75**
mine5 B	5	2310.03	2048.40	**1895.29**

5.5.4 Comparison of the depot-location methods

We compared the three methods for establishing the initial location of depots. We also included in the results, the average cost obtained after testing 12 different scenarios. We tested 4 different initial humidity levels, I_{ij}^{c} : 10%, 30%, 50% and 70% of H_{ij}^{max}. These four scenarios were combined with 3 evaporation scenarios : $c = 1$ shows an approximation of the evaporation behaviour described in Molina-Martínez *et al.* (2006) ; $c = 2$ shows a scenario where there is no evaporation, and the loss of humidity is due to traffic volume only ; $c = 3$ shows a situation where humidity is briefly increased by weather events such as rain. Figure 5.2 shows the behavior of e^{tc} during the 300 time periods for the three scenarios.
Table 5.5 shows the average total cost of the twelve scenarios obtained by each of the three location methods with the main depot at node 0 ($dep = 0$) in columns 3,4 and 5. The same comparison is made when the main depot is located at node n ($dep = n$) in columns 6, 7 and 8. The best solution is highlighted in each case.

The minimum distance method gives the best results in 8 of 10 instances where the main depot is located in node 0 and in 5 out of 10 instances where it is located at node n. In 9 out of 10 instances, better results are obtained when the depot is located at node n.
Table 5.6 shows the average total cost after the ALNS improvement for the twelve scenarios. Columns 3, 4 and 5 compare the three location methods when $dep = 0$, while columns 6, 7 and 8 compare the three methods when $dep - n$. The ALNS performed as many iterations as possible in a time limit of 30 minutes for each scenario. The best solution is highlighted

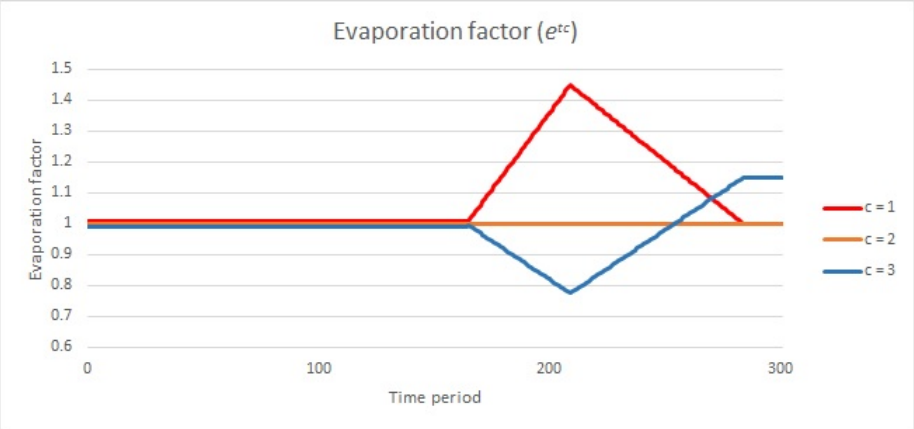

Figure 5.2 Evaporation factor e^{tc} for the three scenarios.

Tableau 5.5 Comparison of the objective function values for the three depot-location methods without ALNS improvement.

Network	Number of vehicles	$dep = 0$			$dep = n$		
		LB	Random	Min. distance	LB	Random	Min. distance
mine1 A	3	1227.53	1229.53	**1200.64**	**1175.97**	1193.47	1213.10
mine1 B	5	1110.36	1112.09	**1098.80**	**1057.09**	1105.54	1076.18
mine2 A	3	3112.55	3144.92	**3078.70**	3138.54	3141.48	**3050.90**
mine2 B	5	2924.49	2967.20	**2909.43**	3018.42	3051.70	**2925.60**
mine3 A	3	5468.26	5465.03	**5456.66**	5440.88	5596.35	**5435.07**
mine3 B	5	**5360.87**	5399.77	5386.89	**5327.36**	5544.08	5344.57
mine4 A	3	3904.26	3912.51	**3884.24**	3877.77	3862.81	**3855.25**
mine4 B	5	3748.94	3768.01	**3742.52**	3704.52	3852.72	**3701.72**
mine5 A	3	2330.84	2310.05	**2302.90**	**2291.42**	2332.02	2326.47
mine5 B	5	2189.67	**2188.90**	2194.57	**2154.91**	2203.68	2197.63

in each case.

After the improvement using the ALNS algorithm, results are similar from Levy and Bodin's method and the minimum distance method. We cannot conclude this behavior is the result of better initial solution, but rather of the random nature of the ALNS operators. Overall, 6 out of 10 instances resulted in a better improvement when the depot is located at node n.

Tableau 5.6 Comparison of the three depot-location methods after the ALNS improvement.

Network	Number of vehicles	$dep = 0$			$dep = n$		
		LB	Random	Min. distance	LB	Random	Min. distance
mine1 A	3	**1024.14**	1060.90	1055.96	**1012.73**	1029.74	1055.23
mine1 B	5	**861.73**	890.61	881.58	902.78	**863.09**	909.46
mine2 A	3	**2721.19**	2927.79	2800.79	2815.71	2856.77	**2707.83**
mine2 B	5	**2428.84**	2781.08	2542.39	2582.77	2654.88	**2437.61**
mine3 A	3	5426.61	5421.49	**5415.62**	5363.95	5488.90	**5402.80**
mine3 B	5	**5293.20**	5321.55	5321.39	**5227.82**	5397.96	5287.01
mine4 A	3	3855.63	3888.48	**3854.57**	3833.20	3844.71	**3821.77**
mine4 B	5	**3708.16**	3749.85	3714.81	3695.59	3836.29	**3674.54**
mine5 A	3	2268.94	2287.28	**2239.83**	**2220.35**	2292.81	2265.29
mine5 B	5	2152.69	2177.87	**2141.99**	**2126.60**	2178.76	2163.66

5.6 Conclusion

The depot location on the network is an important factor to improve dust suppression in the instances tested. In the case of only one depot, it was shown that its location significantly reduces penalty and routing costs. It was also shown that including more than one vehicle significantly reduces operational costs.

Results show the importance of the location of the main depot. In the majority of the instances tested with one or several depots, the total cost was reduced when the main depot, where vehicles start and return at the beginning and end of the time horizon, was located at a node different from 0.

Future work includes finding an algorithm or a procedure to change the initial location of the depots, or, if it is the case reduce or increase the number of depots. After comparing the results among different location methods, it follows that the next step would be to modify the initial placement of depots. However, due to the existance of inventory constraints, any change in the location of depots results in the recalculation of the initial solution and the subsequent improvement. Finding an algorithm to change the depot location is beyond the scope of this work.

Acknowledgements

This project was partially funded by the National Council of Science and Technology (CONACYT) of the Government of Mexico.

CHAPITRE 6

DISCUSSION GÉNÉRALE

Le PCARP avec les contraintes de gestion de stocks est un nouveau problème dans le domaine de tournées sur les arcs. La principale différence par rapport au reste des problèmes de la même classe est que les arêtes du réseau sont considérées comme des clients, chacun avec des exigences en termes de quantité de matériaux requis pour son fonctionnement.

Comme c'est le cas des problèmes de tournées sur les arcs décrits dans la revue de la littérature, pour représenter le problème de l'arrosage de la route dans les mines à ciel ouvert un modèle mathématique a été proposé et utilisé pour trouver la solution optimale des exemplaires de petite taille. Par contre, pour les exemplaires de plus grande taille, on a utilisé un algorithme heuristique pour trouver une solution réalisable dans un temps raisonnable.

Les périodes de temps dont l'horizon de temps est divisé, est une caractéristique qui augmente le nombre de variables du modèle. La taille de l'horizon de temps ainsi que la taille du réseau sont les paramètres à considérer pour établir la taille maximum des exemplaires pour lesquels le modèle est capable de trouver une solution optimale. Lorsqu'un seul véhicule et un seul dépôt sont impliqués, les exemplaires résolus ont moins de 55 arêtes et de 20 à 30 périodes de temps. Cette limite est réduite à 11 arêtes et moins de 18 périodes de temps lorsque plus de véhicules et plus d'un dépôt sont considérés dans le modèle.

Le modèle mathématique pour le PCARP IC est composé des contraintes de capacité et de conservation de flux similaires à ceux qui se trouvent dans d'autres modèles de tournées sur les arcs. Les nouveaux ensembles de contraintes dans ce modèle comprennent la gestion des stocks et le remplissage du camion au dépôt. La gestion des stocks est modélisée par des contraintes qui transferent la quantité dans l'inventaire d'une période de temps à la suivante, au cours de l'horizon de temps. Cela pose une difficulté pour les méthodes heuristiques qui modifient la solution initiale. Tout changement à la solution initiale effectuée par un opérateur représente un changement, ou bien dans le temps de la livraison, ou bien de la quantité livrée à une certaine arête, par conséquent, le reste de la solution doit être calculé de nouveau.

On a étudié différentes variantes du PCARP IC. Tout d'abord, le cas d'un seul véhicule et un seul dépôt a été considéré afin d'introduire le problème. Ensuite, on a étudié le cas

de plus d'un véhicule. Finalement, on a étudié le problème de la localisation de plusieurs dépôts dans un réseau en maintenant les mêmes caractéristiques et contraintes utilisées pour les problèmes de tournées sur les arcs. D'autres variantes du problème ont été proposées pour faire des études futures, comme le cas de la vitesse du véhicule différent pour le service et le *dead-heading* ainsi que le changement de la localisation initial de dépôts.

CHAPITRE 7

CONCLUSION ET RECOMMANDATIONS

Les contributions de cette thèse incluent un modèle mathématique pour le problème d'arrosage des chemins de terre dans les mines à ciel ouvert. Ce modèle est le premier utilisé pour trouver une tournée, qui débute et finit au dépôt, et qui traverse les arêtes du réseau de telle sorte que les coûts de routage et de pénuries sont minimisés. Ce modèle a été modifié pour considérer le cas de plus d'un véhicule et la localisation de plus d'un réservoir d'eau.

Un algorithme ALNS a également été proposé. Il est le premier modèle heuristique pour ce problème.

La localisation de dépôts a également été examinée. On a proposé un algorithme de localisation qui n'avait pas été considéré pour un problème périodique de tournées sur les arcs.

7.1 Synthèse des travaux

Dans ce travail, on a montré comment trouver un ensemble de trajets pour arroser les chemins de terre d'une mine à ciel ouvert de telle sorte que le coût de routage, les coûts de pénurie et, dans un dernier cas, les coûts d'investissement sont minimisés.

Pour la première approche, on développe un modèle mathématique qui est capable de résoudre les réseaux d'une taille de 40 à 55 arêtes, par un nombre réduit de périodes de temps, entre 20 et 30, pour un seul camion et un seul dépôt. Deux solutions ont été explorées, la première avec une quantité constante d'eau et la seconde avec une quantité variable.

Afin de résoudre des problèmes sur des réseaux de plus grande taille, pour un plus grand nombre de périodes de temps et pour plus d'un camion, un algorithme ALNS a été développé. On a été en mesure de proposer une solution réalisable pour les cas de 35 à 55 arêtes pour 300 périodes de temps. Les exemplaires de plus grande taille correspondent à des réseaux réels de mines à ciel ouvert. L'algorithme se compose d'une solution initiale et huit opérateurs de destruction et de réparation qui modifient la solution initiale afin de trouver une amélioration. Les opérateurs ont été testés individuellement et en combinaison avec d'autres

opérateurs. La meilleure combinaison pour chaque ensemble d'exemplaires a été trouvée.

Finalement, pour de réduire encore davantage les coûts, on a essayé de situer plusieurs dépôts le long du réseau. Il a été montré que l'emplacement d'un seul dépôt à un nœud différent a réduit de manière significative le coût de la solution. Trois méthodes de localisation ont été testées pour placer plus d'un dépôt et les résultats ont été rapportés. Les exemplaires testées correspondent à des réseaux réels de mines à ciel ouvert.

7.2 Limitations de la solution proposée

Les limites du modèle mathématique sont liées à la taille des réseaux qu'il est en mesure de résoudre. Quand un seul véhicule est considéré, le modèle est capable de résoudre des problèmes dont les réseaux ont une taille de 40 à 55 arêtes. Lorsque plusieurs éléments sont ajoutés au modèle, tels que plus d'un véhicule ou plus d'un dépôt d'eau, la taille du réseau qu'on est capable de résoudre est réduit à 11 arêtes et moins de 20 périodes de temps.

La limite de l'algorithme ALNS est la taille du réseau qu'il est capable de résoudre. Même si une solution initiale réalisable peut être obtenue dans moins de 5 secondes, le processus d'amélioration n'atteint pas 25000 itérations en moins de 2 heures pour les réseaux de plus de 45 arêtes.

Une limite importante en général est l'utilisation des contraintes de gestion de stock pour modifier une solution. Parce que la quantité en l'inventaire est amenée d'une période de temps à la suivante, il devient difficile de modifier une solution. Chaque modification, indépendamment de l'élément qui a changé, déclenche le calcul de nouveaux trajets dans la tournée et une nouvelle évaluation du coût pour le reste des périodes de temps dans l'horizon. L'utilisation d'algorithmes heuristiques pour modifier une solution initiale est limitée par l'utilisation de contraintes de gestion de stock.

7.3 Améliorations futures

Les améliorations futures consistent à trouver un algorithme pour modifier la localisation initiale des dépôts, ou bien, de modifier le nombre de dépôts dans le réseau une fois qu'ils ont été sélectionnés et une solution a été trouvée. En raison de la limite sur l'utilisation de contraintes de stock expliquée dans la section précédente, la modification de la localisation ou du nombre de dépôts a besoin d'un nouveau calcul complet des routes et des coûts correspondants. L'objectif est de trouver une façon de modifier la solution et de calculer l'effet

d'une telle modification sans la nécessité de la reconstruction de la solution complète.

On a proposé une façon de traiter la différente vitesse des camions avec et sans service. Une amélioration à l'algorithme est d'incorporer cette méthode pour les cas où la vitesse des camions est différente.

LISTE DES RÉFÉRENCES

AMAYA, A., LANGEVIN, A. et TRÉPANIER, M. (2007). The capacitated arc routing problem with refill points. *Operations Research Letters*, 35, 45–53.

AMBERG, A., DOMSCHKE, W. et VOSS, S. (2000). Multiple center capacitated arc routing problems : A tabu search algorithm using capacitated trees. *European Journal of Operational Research*, 124, 360–376.

ASSAD, A. A. et GOLDEN, B. L. (1995). *Chapter 5 Arc routing methods and applications*, Elsevier, vol. Volume 8. 375–483.

BALDACCI, R. et MANIEZZO, V. (2006). Exact methods based on node-routing formulations for undirected arc-routing problems. *Networks*, 47, 52–60.

BARTOLINI, E., CORDEAU, J.-F. et LAPORTE, G. (2013a). An exact algorithm for the capacitated arc routing problem with deadheading demand. *Operations Research*, 61, 315–327.

BARTOLINI, E., CORDEAU, J.-F. et LAPORTE, G. (2013b). Improved lower bounds and exact algorithm for the capacitated arc routing problem. *Mathematical Programming*, 137, 409–452.

BELENGUER, J. M. et BENAVENT, E. (2003). A cutting plane algorithm for the capacitated arc routing problem. *Computers and Operations Research*, 30, 705–728.

BENAVENT, E., CAMPOS, V., CORBERÀN, A. et MOTA, E. (1990). The capacitated arc routing problem. a heuristic algorithm. *Questiió*, 14, 107–122.

BERTAZZI, L. et SPERANZA, M. G. (2012). Inventory routing problems : an introduction. *EURO Journal on Transportation and Logistics*, 1, 307–326.

BEULLENS, P., MUYLDERMANS, L., CATTRYSSE, D. et VAN OUDHEUSDEN, D. (2003). A guided local search heuristic for the capacitated arc routing problem. *European Journal of Operational Research*, 147, 629–643.

BODE, C. et IRNICH, S. (2012). Cut-first branch-and-price-second for the capacitated arc-routing problem. *Operations Research*, 60, 1167–1182.

BRANDAO, J. et EGLESE, R. (2008). A deterministic tabu search algorithm for the capacitated arc routing problem. *Computers and Operations Research*, 35, 1112–1126.

CECALA, A., O'BRIEN, A. D., SCHALL, J., COLINET, J. F., FOX, W. R., FRANTA, R. J., JOY, J., REED, R., REESER, P. W., ROUNDS, J. R. et SCHULTZ, M. J. (2012).

Dust control handbook for industrial minerals mining and processing (no. ri 9689). Report, National Institute for Occupational Safety and Health.

CHAPLEAU, L., FERLAND, J. A., LAPALME, G. et ROUSSEAU, J.-M. (1984). A parallel insert method for the capacitated arc routing problem. *Operations Research Letters*, 3, 95–99.

CHRISTOFIDES, N. (1973). The optimum traversal of a graph. *Omega*, 1, 719–732.

CHU, F., LABADI, N. et PRINS, C. (2003). Lower bounds for the periodic capacitated arc routing problem. *2nd International Workshop on Freight Transportation and Logistics, Odysseus 2003*.

CHU, F., LABADI, N. et PRINS, C. (2005). Heuristics for the periodic capacitated arc routing problem. *Journal of Intelligent Manufacturing*, 16, pp. 243–251.

CHU, F., LABADI, N. et PRINS, C. (2006). A scatter search for the periodic capacitated arc routing problem. *European Journal of Operational Research*, 169, pp. 586–605.

COELHO, L., CORDEAU, J.-F. et LAPORTE, G. (2012a). Consistency in multi-vehicle inventory-routing. *Transportation Research Part C : Emerging Technologies*, 24, pp. 270–287.

COELHO, L., CORDEAU, J.-F. et LAPORTE, G. (2012b). The inventory-routing problem with transshipment. *Computers and Operations Research*, 39, pp. 2537–2548.

CORBERÁN, A. et PRINS, C. (2010). Recent results on arc routing problems : An annotated bibliography. *Networks*, 56, 50–69.

DEL PIA, A. et FILIPPI, C. (2006). A variable neighborhood descent algorithm for a real waste collection problem with mobile depots. *International Transactions in Operational Research*, 13, 125–141.

EGLESE, R. W. (1994). Routeing winter gritting vehicles. *Discrete Applied Mathematics*, 48, 231–244.

EISELT, H. A., GENDREAU, M. et LAPORTE, G. (1995a). Arc routing problems. ii. the rural postman problem. 43, 399–414.

EISELT, H. A., GENDREAU, M. et LAPORTE, G. (1995b). Arc routing problems, part i : the chinese postman problem. *Operations Research*, 43, 231–231. Compilation and indexing terms, Copyright 2013 Elsevier Inc. 1995102879989.

FCM (2005). Dust control for unpaved roads. *Infraguide*, National Research Council and Federation of Canadian Municipalities, vol. 10 de *Infraguide*.

FLEURY, G., LACOMME, P. et PRINS, C. (2004). *Evolutionary Algorithms for Stochastic Arc Routing Problems*, Springer Berlin Heidelberg, vol. 3005 de *Lecture Notes in Computer Science*, book section 51. 501–512.

FLEURY, G., LACOMME, P. et PRINS, C. (2005). Stochastic capacitated arc routing problems. Report, LIMOS/RR-05-12.

GHIANI, G., IMPROTA, G. et LAPORTE, G. (2001). The capacitated arc routing problem with intermediate facilities. *Networks*, 37, 134–143.

GHIANI, G. et LAPORTE, G. (1999). Eulerian location problems. *Networks*, 34, 291–302.

GHIANI, G., MUSMANNO, R., PALETTA, G. et TRIKI, C. (2005). A heuristic for the periodic rural postman problem. *Computers and Operations Research*, 32, pp. 219–228.

GLOVER, F. (1986). Future paths for integer programming and links to artificial intelligence. *Computers and Operations Research*, 13, 533–549.

GOLDEN, B., DEARMON, J. et BAKER, E. (1983). Computational experiments with algorithms for a class of routing problems. *Computers and Operations Research*, 10, pp. 47–59.

GOLDEN, B. et WONG, R. (1981). Capacitated arc routing problems. *Networks*, 11, pp. 305–315.

GREENING, T. (2011). Quantifying the impacts of vehicle-generated dust : A comprehensive approach. Report, The International Bank for Reconstruction and Development / The World Bank, report number 81570.

GREISTORFER, P. (2003). A tabu scatter search metaheuristic for the arc routing problem. *Computers and Industrial Engineering*, 44, 249–266.

HERTZ, A., LAPORTE, G. et HUGO, P. N. (1999). Improvement procedures for the undirected rural postman problem. *INFORMS Journal on Computing*, 11, 53–62.

HERTZ, A., LAPORTE, G. et MITTAZ, M. (2000). A tabu search heuristic for the capacitated arc routing problem. *Operations Research*, 48, 129–135.

HERTZ, A. et MITTAZ, M. (2001). A variable neighborhood descent algorithm for the undirected capacitated arc routing problem. *Transportation Science*, 35, 425–434.

HIRABAYASHI, R., SARUWATARI, Y. et NISHIDA, N. (1992). Tour construction algorithm for the capacitated arc routing problems. *Asia-Pacific Journal of Operational Research*, 9, 155–175.

HONGTAO, H., TIANTANG, L., NING, Z., YITING, Z. et DEQUAN, M. (2013). A hybrid genetic algorithm with perturbation for the multi-depot capacitated arc routing problem. *Journal of Applied Sciences*, 13, 3239–3244.

KANSOU, A. et YASSINE, A. (2009). Ant colony system for the periodic capacitated arc routing problem. *Proc. 2009 International Network Optimization Conference*. pp. 1–7.

KIRLIK, G. et SIPAHIOGLU, A. (2012). Capacitated arc routing problem with deadheading demands. *Computers and Operations Research*, 39, 2380–2394.

LACOMME, P., PRINS, C. et RAMDANE-CHÉRIF, W. (2001). *A Genetic Algorithm for the Capacitated Arc Routing Problem and Its Extensions*, Springer Berlin Heidelberg, vol. 2037 de *Lecture Notes in Computer Science*, book section 49. 473–483.

LACOMME, P., PRINS, C. et RAMDANE-CHÉRIF, W. (2002). Evolutionary algorithms for multiperiod arc routing problems. *Proc. of the 9th Int. Conf. on Information Processing and Management of Uncertainty in Knowledge-Based systems (IPMU 2002)*. pp. 845–852.

LACOMME, P., PRINS, C. et RAMDANE-CHÉRIF, W. (2005). Evolutionary algorithms for periodic arc routing problems. *European Journal of Operational Research*, 165, pp. 535–553.

LAPORTE, G., MUSMANNO, R. et VOCATURO, F. (2010). An adaptive large neighbourhood search heuristic for the capacitated arc-routing problem with stochastic demands. *Transportation Science*, 44, pp. 125–135.

LAPORTE, G. et OSMAN, I. H. (1995). Routing problems : A bibliography. *Annals of Operations Research*, 61, 227–262.

LEVY, L. et BODIN, L. (1989). The arc oriented location routing problem. *INFOR*, 27, 74 – 94.

LI, J.-Q., MIRCHANDANI, P. et KNIGHTS, P. (2008). Water truck routing and location of refilling stations in open pit mines. *Proceedings of 2008 Australian Mining Technology Conference*. The Australian Institute of Mining and Metallurgy, Sunshine Coast, Australia, pp. 141–156.

LIU, M. et RAY, T. (2012). A memetic algorithm with random key crossover and modified neighborhood search for the solution of capacitated arc routing problems. *2012 6th International Conference on Genetic and Evolutionary Computing, ICGEC 2012, August 25, 2012 - August 28, 2012*. IEEE Computer Society, Proceedings - 2012 6th International Conference on Genetic and Evolutionary Computing, ICGEC 2012, 433–436.

LIU, T., JIANG, Z. et GENG, N. (2012). A genetic local search algorithm for the multi-depot heterogeneous fleet capacitated arc routing problem. *Flexible Services and Manufacturing Journal*, 1–25.

LIU, T., JIANG, Z. et GENG, N. (2013). A memetic algorithm with iterated local search for the capacitated arc routing problem. *International Journal of Production Research*, 51, 3075–3084.

LONGO, H., DE ARAGÃO, M. P. et UCHOA, E. (2006). Solving capacitated arc routing problems using a transformation to the cvrp. *Computers and Operations Research*, 33, 1823–1837.

LOPES, R. B., FERREIRA, C., SANTOS, B. S. et BARRETO, S. (2013). A taxonomical analysis, current methods and objectives on location-routing problems. *International Transactions in Operational Research*, 20, 795–822.

LOPES, R. B., PLASTRIA, F., FERREIRA, C. et SANTOS, B. S. (2014). Location-arc routing problem : Heuristic approaches and test instances. *Computers and Operations Research*, 43, 309–317.

MARTÍ, R., LAGUNA, M. et GLOVER, F. (2006). Principles of scatter search. *European Journal of Operational Research*, 169, 359–372.

MARTIN, A., OWENDE, P., WARD, S. et O'MAHONY, M. (2000). A timber extraction method based on pavement serviceability and forest inventory data. *Forest Science*, 46, 76–85.

MARTINELLI, R., POGGI, M. et SUBRAMANIAN, A. (2013). Improved bounds for large scale capacitated arc routing problem. *Computers and Operations Research*, 40, 2145–2160.

MARZOLF, F., TREÉPANIER, M. et LANGEVIN, A. (2006). Road network monitoring : algorithms and a case study. *Computers and Operations Research*, 33, pp. 3494–3507.

MEI, Y., TANG, K. et YAO, X. (2010). Capacitated arc routing problem in uncertain environments. *2010 6th IEEE World Congress on Computational Intelligence, WCCI 2010 - 2010 IEEE Congress on Evolutionary Computation, CEC 2010, July 18, 2010 - July 23, 2010*. IEEE Computer Society, 2010 IEEE World Congress on Computational Intelligence, WCCI 2010 - 2010 IEEE Congress on Evolutionary Computation, CEC 2010.

MEI, Y., TANG, K. et YAO, X. (2011). A memetic algorithm for periodic capacitated arc routing problem. *Systems, Man, and Cybernetics, Part B : Cybernetics, IEEE Transactions on*, PP, pp. 1–14.

MOLINA-MARTÍNEZ, J., MARTÍNEZ-ALVAREZ, V., GONZÁLEZ-REAL, M. et BAILLE, A. (2006). A simulation model for predicting hourly pan evaporation from meteorological data. *Journal of Hydrology*, 318, pp. 250–261.

MONROY, I., AMAYA, C. et LANGEVIN, A. (2011). The periodic capacitated arc routing problem with irregular services. *Discrete Applied Mathematics*, 161, pp. 691–701.

MOREIRA, L. M., OLIVEIRA, J. F., GOMES, A. M. et FERREIRA, J. S. (2007). Heuristics for a dynamic rural postman problem. *Computers and Operations Research*, 34, 3281–3294.

MULLASERIL, P. A. (1997). *Capacitated rural postman problem with time windows and split delivery*. Thesis.

NEULICHT, R. et SHULAR, J. (1998). Emission factor documentation for ap-42. section 13.2.2. unpaved roads. Rapport technique, U.S. Environmental Protection Agency.

PEARN, W.-L., ASSAD, A. et GOLDEN, B. L. (1987). Transforming arc routing into node routing problems. *Computers and Operations Research*, 14, 285–288.

PERRIER, N., LANGEVIN, A. et AMAYA, C.-A. (2008). Vehicle routing for urban snow plowing operations. *Transportation Science*, 42, 44–56.

POLACEK, M., DOERNER, K. F., HARTL, R. F. et MANIEZZO, V. (2008). A variable neighborhood search for the capacitated arc routing problem with intermediate facilities. *Journal of Heuristics*, 14, 405–423.

PRODHON, C. (2011). A hybrid evolutionary algorithm for the periodic location-routing problem. *European Journal of Operational Research*, 210, 204–212.

PRODHON, C. et PRINS, C. (2008). *A Memetic Algorithm with Population Management (MA|PM) for the Periodic Location-Routing Problem*, Springer Berlin Heidelberg, vol. 5296 de *Lecture Notes in Computer Science*, book section 4. 43–57.

RIQUELME RODRÍGUEZ, J. P., GAMACHE, M. et LANGEVIN, A. (2013). Periodic capacitated arc-routing problem with inventory constraints. *Journal of the Oparational Research Society*.

RIQUELME RODRÍGUEZ, J. P., GAMACHE, M. et LANGEVIN, A. (2014). Adaptive large neighborhood search for the periodic capacitated arc routing problem with inventory constraints. Report, HEC Montreal, Report G-2014-01.

ROPKE, S. et PISINGER, D. (2006). An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows. *Transportation Science*, 40, 455–472.

SALAZAR-AGUILAR, M., LANGEVIN, A. et LAPORTE, G. (2012). Synchronized arc routing for snow plowing operations. *Computers and Operations Research*, 39, 1432–1440.

SALAZAR-AGUILAR, M., LANGEVIN, A. et LAPORTE, G. (2013). The synchronized arc and node routing problem : Application to road marking. *Computers and Operations Research*, 40, 1708–1715.

SBIHI, A. et EGLESE, R. W. (2010). Combinatorial optimization and green logistics. *Annals of Operations Research*, 175, 159–175.

SHAW, P. (1997). A new local search algorithm providing high quality solutions to vehicle routing problems. *APES Group, Dept of Computer Science, University of Strathclyde, Glasgow, Scotland, UK*.

SILVER, E.-A., PYKE, D.-F. et PETERSON, R. (1998). *Inventory Management and Production Planning and Scheduling*. John Wiley and Sons.

STERN, H. I. et DROR, M. (1979). Routing electric meter readers. *Computers and Operations Research*, 6, 209–223.

TAGMOUTI, M., GENDREAU, M. et POTVIN, J.-Y. (2007). Arc routing problems with time-dependent service costs. *European Journal of Operational Research*, 181, 30–39.

THOMPSON, R.-J. et VISSER, A.-T. (2000). Mining roads. mine haul road design, construction and maintenance management. *The Journal of the South African institute of mining and metalurgy*, 100, 169–180.

TIAN, G., ZHANG, L. et QU, F. (1996). *A study using a near infrared light scattering method to monitor coal mine respirable dust concentration*, Balkema Publishers, Rotterdam. 245–250.

ULUSOY, G. (1985). The fleet size and mix problem for capacitated arc routing. *European Journal of Operational Research*, 22, 329–337.

USBERTI, F. L., FRANCA, P. M. et FRANCA, A. L. M. (2011). The open capacitated arc routing problem. *Computers and Operations Research*, 38, 1543–1555.

WOHLK, S. (2005). *Contributions to arc routing*. Thesis.

WOHLK, S. (2008). *A Decade of Capacitated Arc Routing*, Springer US, vol. 43 de *Operations Research/Computer Science Interfaces*, book section 2. 29–48.