

A Branch and Cut Approach for the Mixed Capacitated General Routing Problem

Adamo Bosco

November 2010



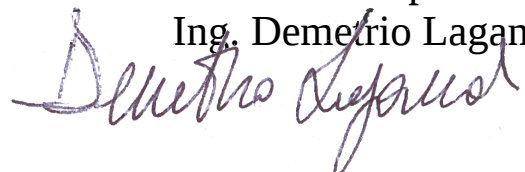
Department of Electronics, Informatics and Systems
University of Calabria

A thesis submitted for the degree of
PhD in Operations Research
SSD: MAT/09

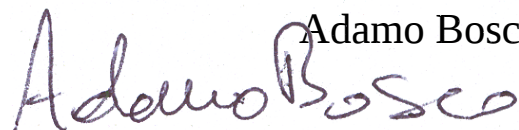
Coordinator
Prof. Lucio Grandinetti



Supervisor
Ing. Demetrio Laganà



Candidate
Adamo Bosco



Un approccio Branch and Cut per il Problema capacitivo di instradamento su grafi misti con servizio sui nodi, sugli spigoli e sugli archi

Adamo Bosco

Novembre 2010



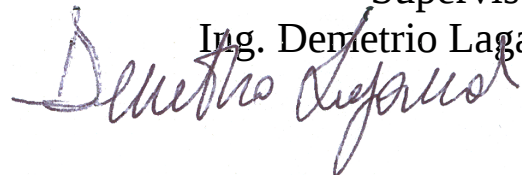
Dipartimento di Elettronica, Informatica e Sistemistica
Università della Calabria

*Tesi sottomessa per il dottorato di ricerca in
Ricerca Operativa
SSD: MAT/09*

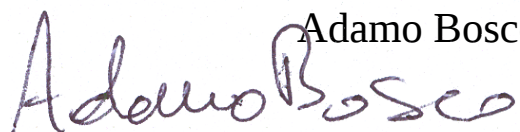
Coordinatore
Prof. Lucio Grandinetti



Supervisore
Ing. Demetrio Laganà



Candidato
Adamo Bosco



Abstract

The issue addressed in this thesis consists in modeling and solving the Mixed Capacitated General Routing Problem (MCGRP). This problem generalizes many routing problems, either in the Node or in the Arc routing family. This makes the problem a very general one and gives it a big interest in real-world applications. Despite this, and because of the native difficulty of the problem, very few papers have been devoted to this argument. In the thesis an integer programming model for the MCGRP is proposed and several valid inequalities for the undirected Capacitated Arc Routing polyhedron are extended and generalized to the MCGR polyhedron. A branch and cut algorithm for the MCGRP is developed and tested on a dataset of new instances derived from mixed CARP benchmark instances. Moreover an heuristic procedure is defined in order to find a good upper bound aimed at helping the branch and cut algorithm to cut off unpromising regions of the search tree. This schema will be used and extended in future works to solve bigger real-world instances. Extensive numerical experiments indicate that the algorithm is able to optimally solve many instances. In general, it provides valid lower and upper bounds for the problem in a reasonable amount of time.

Sommario

Nell'ambito della tesi di dottorato sono stati studiati i problemi di instradamento con vincoli capacitivi su grafi misti, noti nella letteratura scientifica con l'acronimo MCGRPs (Mixed Capacitated General Routing Problems).

Si tratta di problemi che presentano un forte impatto applicativo in una grande varietà di situazioni reali: la manutenzione delle strade, la raccolta dei rifiuti, la consegna della posta, l'instradamento dei bus a servizio delle scuole, etc. Esempi classici in cui è possibile riscontrare problemi di questa natura ricorrono nella progettazione di rotte per la raccolta dei rifiuti urbani. In questi casi, la richiesta di servizio nelle abitazioni lungo le strade può essere modellata come domanda di servizio distribuita su un sottoinsieme di archi e/o spigoli del grafo, mentre il servizio concentrato in punti isolati o in grossi centri di accumulo (ospedali, scuole, supermercati, ...) può essere modellato come domanda di servizio in un sottoinsieme di nodi del grafo.

Si assume che il servizio sia eseguito con una flotta costituita da un numero limitato di veicoli aventi la stessa capacità (flotta omogenea), che partono dal deposito e vi ritornano dopo avere servito alcuni archi, spigoli e/o nodi del grafo. La rete stradale è rappresentata da un grafo misto, costituito da archi (collegamenti orientati, rappresentanti strade a senso unico), spigoli (collegamenti non orientati, rappresentanti strade a doppio senso) e nodi (rappresentanti punti di intersezione dei collegamenti, punti di domanda ed il deposito).

Tale problema è NP-arduo in quanto riducibile polinomialmente ad altri problemi NP-ardui. In particolare, nel caso in cui: a) la flotta sia costituita da un solo veicolo la cui capacità sia non minore della somma delle domande di servizio; b) il grafo sia non orientato (nessun arco) e c) il servizio sia solo su un sottoinsieme di spigoli; allora il MCGRP si riduce al cosiddetto URPP

(Undirected Rural Postman Problem), che é stato dimostrato essere un problema NP-arduo (Lenstra and Kan (1976)). Un gran numero di problemi di instradamento con servizio solo su archi e spigoli, noti in letteratura come ARPs (Arc Routing Problems), sono varianti del problema MCGRP, ad esempio il problema del postino cinese capacitivo (CCPP), il problema del postino rurale (RPP), in cui solo alcuni collegamenti devono essere serviti e la corrispondente versione capacitiva (CRPP). In alcuni problemi il servizio é richiesto sia su nodi che su collegamenti tra nodi, il problema risultante é chiamato General Routing Problem (GRP) e, nella sua versione capacitiva, é noto come Capacitated GRP (CGRP). Il problema oggetto della tesi é un'ulteriore estensione naturale di questo problema, cioé il CGRP su grafi misti.

La letteratura sui problemi di routing generalmente distingue problemi di node routing e problemi di arc routing, a seconda che il servizio é richiesto, rispettivamente, su nodi o archi (tutti o una parte). In teoria i problemi di arc routing possono essere convertiti in problemi di node routing equivalenti (Pearn et al. (1987), Longo et al. (2006), Baldacci and Maniezzo (2006)). Tuttavia questa trasformazione causa un incremento della dimensione delle istanze, di conseguenza, soprattutto negli ultimi anni, molti ricercatori hanno privilegiato lo studio diretto dei problemi di arc routing.

Letchford, Corberán and Sanchis sono stati i primi a lavorare sui problemi di general routing su grafi non orientati. In particolare Letchford (1996), Letchford (1999) e Corberán and Sanchis (1998) hanno proposto una classe di disequaglianze valide per il poliedro del GRP, successivamente Corberán et al. (2001) ha descritto un algoritmo di tipo cutting plain per il problema. Muyldermans et al. (2005) ha descritto come alcune procedure di local search dei problemi di node routing possano essere adattate al GRP.

Passando al problema di general routing su grafi misti (MGRP), il primo contributo significativo che si trova in letteratura é dovuto a Corberán et al. (2003). Gli autori hanno proposto una formulazione matematica del problema molto stringente, tramite la quale sono riusciti a fornire una descrizione molto raffinata del poliedro del MGRP. Gli stessi hanno progettato

una algoritmo cutting plane, esteso poi da Corberán et al. (2005) tramite l'introduzione di nuove famiglie di disequaglianze valide inducenti faccette e relative procedure di separazione.

Riguardo al problema MCGRP, in letteratura non esistono approcci esatti. Pandit and Muralidharan (1995) hanno proposto una procedura euristica, di tipo instrada prima e separa dopo, che costruisce prima un giant tour contenente tutti gli elementi di servizio e quindi partiziona questo tour in sotto-tour che rispettano il vincolo di capacità. Gutiérrez et al. (2002) hanno introdotto una procedura alternativa di tipo separa prima e instrada dopo.

Nell'ambito della tesi é stato definito un algoritmo esatto in grado di risolvere all'ottimo istanze di medie dimensioni; inoltre, é stata sviluppata una procedura euristica composta da una fase costruttiva e da una fase migliorativa (local search) in grado sia di fornire un buon upper bound all'algoritmo esatto e sia di determinare una buona soluzione ammissibile laddove il metodo esatto dovesse fallire. La soluzione a questo tipo di problemi ha una serie di impatti pratici molto importanti, se si pensa che esistono molteplici tipi di applicazioni reali che possono essere modellate come MCGRP Lenstra and Kan (1976). Una soluzione ottima, o near-ottima, del problema di instradamento puó avere un impatto pratico molto rilevante se si pensa che i costi di trasporto incidono in modo significativo sui costi complessivi delle aziende che offrono servizi logistici di tipo distributivo.

To my parents

Contents

List of Figures	v
List of Tables	vii
List of Algorithms	ix
1 Introduction	1
1.1 Literature review	3
1.1.1 The General Routing Problem	3
1.1.2 The Mixed General Routing Problem	3
1.1.3 The Mixed Capacitated General Routing Problem	3
1.2 Contributions	4
2 Problem definition	5
2.1 Notations	5
2.2 Mathematical formulation	7
3 Valid inequalities and separation algorithms	13
3.1 Connectivity constraints separation	13
3.1.1 Heuristic Algorithm	13
3.1.2 Exact Algorithm	16
3.2 Surrogate inequalities	17
3.2.1 Capacity and odd-edge inequalities	18
3.2.2 Capacity inequalities separation	19
3.2.3 Odd-edge inequalities separation	21

CONTENTS

4	An Upper bound	27
4.1	The initial solution	27
4.1.1	Partition-First-Route-Next	28
4.1.2	Clustering	33
4.1.2.1	Metric	34
4.1.2.2	Clusters generation	34
4.2	Local Search	36
5	The B&C Algorithm	39
5.1	Cut pool management	39
5.2	Root node generation	39
5.3	The Branch and Cut algorithm	40
6	Computational experiments	43
6.1	A First Illustration	43
6.2	The Instances	44
6.3	Numerical Results	45
7	Conclusions	51
	References	53

List of Figures

2.1	Flow constraints illustration	8
2.2	Connectivity constraints illustration	9
4.1	Computation of the average dinstance between service elements.	34
6.1	$G = (V, E, A)$	44
6.2	Optimal solution.	44
6.3	Comparison among Lower Bound, Upper Bound and final cost - Instances mgval1A to mgval5D	48
6.4	Comparison among Lower Bound, Upper Bound and final cost - Instances mgval6A to mgval10D	49

LIST OF FIGURES

List of Tables

6.1	Computational results for <i>mgval</i> instances	47
-----	--	----

LIST OF TABLES

List of Algorithms

1	Heuristic separation for the connectivity inequalities	14
2	Exact connectivity inequalities separation	16
3	Capacity inequalities separation	21
4	Perturbed demand procedure	21
5	Cut-tree	23
6	Exact Odd Edge cut separation	24
7	Heuristic	27
8	Initial solution	27
9	Partition-First-Route-Next	29
10	Clustering	34
11	LocalSearch	36
12	Swap	37
13	Subsets generation	39
14	Branch & Cut	40

LIST OF ALGORITHMS

1

Introduction

The thesis deals with the problem of finding the routes for a set of vehicles in order to ensure a service inside a given area. Each vehicle starts from an unique depot and comes back to it after a service trip. The streets are modeled by edges or arcs of a network whose nodes represent the intersection points of the streets or some points of interest. In particular, a one way street is modeled with an arc in the network and a two way street with an edge.

The problem of finding a minimum distance tour that covers at least once each street of a network is known in literature as Chinese Postman Problem (CPP). All other arc routing problems (ARP_s) are variants of it. For example if the vehicle has a limited capacity and the streets have a demand (or equivalently each tour has a time limit and a known traversal time for each street) the resulting problem is called Capacitated Chinese Postman Problem (CCPP). In another variant only some streets need a service and we talk about Rural Postman Problem (RPP). If we have also capacity or time limitations we obtain the Capacitated Rural Postman Problem (CRPP). Sometimes a service is required on some nodes as well as on some streets and the resulting problem is called General Routing Problem (GRP). The capacitated version is the Capacitated General Routing Problem (CGRP).

The literature traditionally devoted to vehicle routing problems considers two different classes of problems: node routing problems (NRP_s) and arc routing problems (ARP_s). In NRP_s the service activity occurs at all (or at some subset of) the nodes, whereas in ARP_s a single vehicle or a fleet of vehicles services all (or some subset of) edges and/or arcs. Although NRP_s have been studied more extensively, ARP_s have

1. INTRODUCTION

raised a growing interest in the two last decades. Theoretically an ARP can be converted into an equivalent NRP (see, e.g., Pearn et al. (1987), Longo et al. (2006), and Baldacci and Maniezzo (2006)). However the transformation increases the size of the instances to be solved. Consequently, most researchers prefer to study the ARP_s directly. Despite the success of exact and heuristic methods for NRP_s and ARP_s , most models proposed in the literature do not give a suitable representation of real scenarios.

The recent literature refers to a more general and effective class of problems, where the service activity occurs both at all (or at some subset of) the nodes and at all (or at some subset of) arcs and/or edges. Such problems are denoted as General Routing Problems (GRP_s). Several applications arising in the vehicle routing field may be naturally modeled as GRP (Orloff (1974), Lenstra and Kan (1976)). For example, in designing routes in an urban waste collection context, the collection along a street may be modeled by means of required arcs or edges, whereas the collection occurring in specific points (e.g., hospitals, schools, supermarkets, and multi-story apartment blocks) may be modeled by means of required nodes. Similarly, in the postal delivery services, a set of customers in the same neighborhood with high demands may be modeled by means of required nodes, whereas a set of customers in the same street with soft demands may be modeled by means of required arcs/edges.

This paper refers to the Mixed Capacitated General Routing Problem (MCGRP), i.e., the problem of finding a set of vehicle routes on a mixed graph such that each route starts and ends at the depot, each required node/arc/edge is serviced by exactly one vehicle, the total demand serviced by each vehicle does not exceed its capacity, and the total traveling cost is minimized. This problem is also referred in the literature as CGRP, CGRP-m (Capacitated General Routing Problem on mixed graphs) or NEARP (Node, Edge and Arc Routing Problem). Many routing problems are special cases of the MCGRP, like the capacitated ARP (Golden and Wong (1981)), the capacitated NRP or capacitated vehicle routing problem (see Cornuejols and Harche (1993)), and the above-mentioned GRP and MGRP. Since the MCGRP includes a large side of NP-hard problems, it is also an NP-hard problem. To our knowledge, the MCGRP was exclusively tackled by means of heuristic approaches.

1.1 Literature review

1.1.1 The General Routing Problem

Letchford, Corberán and Sanchis firstly worked on the GRP defined over an undirect graph. Specifically, Letchford (1996), Letchford (1999) and Corberán and Sanchis (1998) proposed a large class of valid inequalities for the GRP polyhedron. Then Corberán et al. (2001) described a cutting plane algorithm for the GRP based on facet-inducing inequalities. Other theoretical results were presented by Reinelt and Theis (2008), whereas Muyldermans et al. (2005) described how the well-known 2-opt and 3-opt local search procedures for NRP_s can be adapted to tackle ARP_s and GRP_s .

1.1.2 The Mixed General Routing Problem

The first remarkable contribution focused on the GRP over a mixed graph (MGRP) was proposed by Corberán et al. (2003). They presented an integer programming formulation and a partial description of the related polyhedron. Furthermore, they reported computational results obtained by performing a cutting-plane algorithm. Corberán et al. (2005) considerably improved this algorithm, by defining a new family of facet-defining inequalities and new separation procedures. The algorithm, designed for the MGRP, has been used as a black-box solver to tackle real instances of the asymmetric travelling salesman problem with time-dependent costs (see Albiach et al. (2008)). Soler et al. (2008) studied a generalization of the MGRP with turn penalties and forbidden turns. The MGRP problem may be also modeled by resorting to windy graphs. For example Corberán et al. (2007) and Corberán et al. (2008) presented a strong windy general routing polyhedron description and designed a branch and cut algorithm able to optimally solve a quite large number of MGRP benchmark instances.

1.1.3 The Mixed Capacitated General Routing Problem

Pandit and Muralidharan (1995) proposed a heuristic procedure named ROUTE1 that starts with a condensed sub-graph obtained from the original network by considering only the required arcs, edges and nodes. Since the condensed sub-graph is generally disconnected, the connection is reached by adding to it the cheapest paths linking two nodes of disjoint connected components. The sub-graph is then converted into an Eulerian which admits a giant tour. A feasible MCGRP is obtained by cutting the giant

1. INTRODUCTION

tour into smaller tours satisfying the capacity constraints. Gutiérrez et al. (2002) introduced an alternative procedure, based on the partition-first-route-next paradigm, improving previous results. Finally, Prins and Bouchenoua (2005) described a memetic algorithm for the MCGRP. Three simple procedures, named *nearest neighbor heuristic*, *merge heuristic* and *tour splitting heuristic*, were defined to initialize the memetic algorithm.

1.2 Contributions

In this section, we list the major contributions of this thesis.

- We develop a linear integer programming model for the MCGRP, based on three-index link variables and two-index node variables. A branch-and-cut algorithm is designed to optimally solve a class of hard instances.
- Several well-known valid inequalities for the Undirected Capacitated Arc Routing polyhedron are extended to the Mixed Capacitated General Routing polyhedron and the relevant separation procedures are adapted to cut off infeasible MCGRP solutions.
- A tight lower bound at the root node of the search tree is obtained by strengthening the linear relaxation of the MCGRP through a set of connectivity constraints and valid inequalities that are conveniently generated and inserted into the initial formulation.
- An upper bound procedure is developed in order to find a good feasible solution of the MCGRP aimed at reaching an efficient pruning in the search tree.

2

Problem definition

2.1 Notations

Let $G = (V, A, E)$ be a mixed graph defined by a set of nodes V , a set of arcs A and a set of edges E . Each entity in $A \cup E$ will be referred in the following as *link*. A non-negative cost c_{ij} is associated to each link (i, j) . Node $1 \in V$ represents the depot at which identical vehicles of capacity Q are based. Some entities (*tasks*) require a service, i.e., need to be visited by a vehicle. Specifically, setting $C = V \setminus \{1\}$, $C_R \subseteq C$ denotes the set of required nodes (or *node-tasks*), while $A_R \subseteq A$ and $E_R \subseteq E$ denote the set of required arcs (or *arc-tasks*) and required edges (or *edge-tasks*), respectively. A demand $q_i > 0$ is associated to each required node $i \in C_R$, and a demand $d_{ij} > 0$ is associated to each required link $(i, j) \in A_R \cup E_R$. Let $K = \{1, \dots, m\}$ be the set of vehicles, where $m \geq \eta$. In order to find η , i.e., the minimum number of vehicles servicing all tasks, a one-dimensional bin packing problem is solved Coffman et al. (1996). A simple lower bound on η is defined by $\lceil (\sum_{(i,j) \in E_R \cup A_R} d_{ij} + \sum_{i \in C_R} q_i) \backslash Q \rceil$. Tasks cannot be preempted, i.e., each task must be serviced by exactly a vehicle. To ensure feasibility, we assume that no demand associated to tasks exceeds Q . Moreover, let $S \subset V$ be a subset of nodes, and \bar{S} its complementary set with respect to V ($\bar{S} = V \setminus S$).

The sets defined below are referred in the rest of the paper as *cutsets* and depend on S .

- $A^+(S) = \{(i, j) \in A, i \in S, j \in \bar{S}\} = A(S : \bar{S})$ is the set of arcs leaving S .
- $A^-(S) = \{(i, j) \in A, i \in \bar{S}, j \in S\} = A(\bar{S} : S)$ is the set of arcs entering S .

2. PROBLEM DEFINITION

- $E^+(S) = \{(i, j) \in E, i \in S, j \in \bar{S}\} = E(S : \bar{S})$ is the set of edges between S and \bar{S} which are traversed from i to j in the solution.
- $E^-(S) = \{(i, j) \in E, i \in \bar{S}, j \in S\} = E(\bar{S} : S)$ is the set of edges between S and \bar{S} which are traversed from i to j in the solution.
- $A_R^+(S) = \{(i, j) \in A_R, i \in S, j \in \bar{S}\} = A_R(S : \bar{S})$ is the set of arc-tasks leaving S .
- $A_R^-(S) = \{(i, j) \in A_R, i \in \bar{S}, j \in S\} = A_R(\bar{S} : S)$ is the set of arc-tasks entering S .
- $E_R^+(S) = \{(i, j) \in E_R, i \in S, j \in \bar{S}\} = E_R(S : \bar{S})$ is the set of edge-tasks between S and \bar{S} which are traversed from i to j in the solution.
- $E_R^-(S) = \{(i, j) \in E_R, i \in \bar{S}, j \in S\} = E_R(\bar{S} : S)$ is the set of edge-tasks between S and \bar{S} which are traversed from i to j in the solution.
- $A(S) = A^+(S) \cup A^-(S)$ is the set of arcs leaving or entering S .
- $E(S) = E^+(S) \cup E^-(S)$ is the set of edges between S and \bar{S} .
- $A_R(S) = A_R^+(S) \cup A_R^-(S)$ is the set of arc-tasks leaving or entering S .
- $E_R(S) = E_R^+(S) \cup E_R^-(S)$ is the set of edge-tasks between S and \bar{S} .

When $S = \{i\}$ the preceding notations become A_i^+ , A_i^- , E_i^+ , E_i^- , A_{Ri}^+ , A_{Ri}^- , E_{Ri}^+ , E_{Ri}^- , A_i , E_i , A_{Ri} , and E_{Ri} . Moreover, we define the following sets depending on S .

- $S_R = S \cap C_R$ is the set of node-tasks in S .
- $\gamma_R(S) = \{(i, j) \in A_R \cup E_R : i \in S, j \in S\}$ is the set of arc-tasks and edge-tasks with both endpoints in S .

Finally, let $G^R = (V_R, E_R, A_R)$ be the graph induced on G by all the required links and nodes. Observe that V_R may contain only one required node. G^R is, in general, non-connected. Let t be the number of its connected components and V_1, V_2, \dots, V_t their related node sets (*R-sets*). The subgraphs of G induced by the R-sets define the so-called *R-connected* components of G .

2.2 Mathematical formulation

In this section we propose an integer linear programming formulation based on three-index link variables and two-index node variables.

The following variables indicate which vehicle serves each required entity (*service variables*):

$$x_{ij}^k = \begin{cases} 1 & \text{if and only if } (i, j) \in A_R \cup E_R \text{ is serviced by } k \in K \\ & \text{which travels from node } i \text{ to node } j; \\ 0 & \text{otherwise.} \end{cases}$$

$$z_i^k = \begin{cases} 1 & \text{if and only if } i \in C_R \text{ is serviced by } k \in K; \\ 0 & \text{otherwise.} \end{cases}$$

Moreover each link can be deadheaded, i.e., traversed without being serviced, any number of times. Then we introduce the following variables:

$$y_{ij}^k = \text{number of deadheading of } (i, j) \in A \cup E \text{ from node } i \\ \text{to node } j \text{ by } k \in K.$$

The objective is the minimization of the total cost:

$$\begin{aligned} \text{Minimize } \lambda = & \sum_{k \in K} \sum_{(i,j) \in E_R} c_{ij} (x_{ij}^k + x_{ji}^k) + \sum_{k \in K} \sum_{(i,j) \in A_R} c_{ij} x_{ij}^k + \\ & \sum_{k \in K} \sum_{(i,j) \in E} c_{ij} (y_{ij}^k + y_{ji}^k) + \sum_{k \in K} \sum_{(i,j) \in A} c_{ij} y_{ij}^k \end{aligned} \quad (2.1)$$

The constraints can be divided in four categories. Firstly each required entity must be serviced by exactly one vehicle (*assignment constraints*):

$$\sum_{k \in K} (x_{ij}^k + x_{ji}^k) = 1, \forall (i, j) \in E_R \quad (2.2)$$

$$\sum_{k \in K} x_{ij}^k = 1, \forall (i, j) \in A_R \quad (2.3)$$

2. PROBLEM DEFINITION

$$\sum_{k \in K} z_i^k = 1, \forall i \in C_R \quad (2.4)$$

The total demand serviced by each vehicle must not exceed its capacity (*knapsack constraints*):

$$\sum_{(i,j) \in E_R} d_{ij}(x_{ij}^k + x_{ji}^k) + \sum_{(i,j) \in A_R} d_{ij}x_{ij}^k + \sum_{i \in C_R} q_i z_i^k \leq Q, \forall k \in K \quad (2.5)$$

The following inequalities represent the *flow constraints* and model at the same time the *balanced set* constraints and the *parity constraints* at each node of the graph, including the depot:

$$\begin{aligned} \sum_{j:(i,j) \in A_{Ri}^+} x_{ij}^k + \sum_{j:(i,j) \in A_i^+} y_{ij}^k - \sum_{j:(j,i) \in A_{Ri}^-} x_{ji}^k - \sum_{j:(j,i) \in A_i^-} y_{ji}^k = \\ \sum_{j:(j,i) \in E_{Ri}^-} x_{ji}^k + \sum_{j:(j,i) \in E_i^-} y_{ji}^k - \sum_{j:(i,j) \in E_{Ri}^+} x_{ij}^k - \sum_{j:(i,j) \in E_i^+} y_{ij}^k, \end{aligned} \quad (2.6)$$

$\forall k \in K, i \in V$

For a graphical explanation of these constraints consider the node 3 in the figure 2.1: The parity constraints impose that the number of node's crossing must be even, while

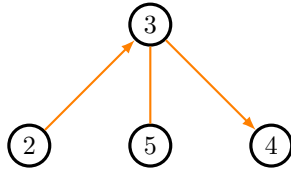


Figure 2.1: Flow constraints illustration

the balanced constraints impose that the difference between the number of arcs outgoing from the node and the number of arcs incoming in the node must be less than or equal to the number of edges incident to it.

The following constraints (*connectivity constraints*) impose that for each subset of nodes (excluding the depot) with a required link inside that is serviced by a vehicle, at least two links incident to the subset must be used to visit it (deadheaded or serviced);

2.2 Mathematical formulation

they also eliminate subtours disjoint from the depot.

$$\begin{aligned}
 & \sum_{\forall(i,j) \in E_R^+(S)} x_{ij}^k + \sum_{\forall(j,i) \in E_R^-(S)} x_{ji}^k + \sum_{\forall(i,j) \in A_R^+(S)} x_{ij}^k + \sum_{\forall(j,i) \in A_R^-(S)} x_{ji}^k + \quad (2.7) \\
 & \sum_{\forall(i,j) \in E^+(S)} y_{ij}^k + \sum_{\forall(j,i) \in E^-(S)} y_{ji}^k + \sum_{\forall(i,j) \in A^+(S)} y_{ij}^k + \sum_{\forall(j,i) \in A^-(S)} y_{ji}^k \geq \\
 & \begin{cases} 2(x_{uv}^k + x_{vu}^k) & \text{if } (u,v) \in \gamma_R(S) \text{ is an edge-task;} \\ 2x_{uv}^k & \text{if } (u,v) \in \gamma_R(S) \text{ is an arc-task;} \\ 2z_h^k & \text{if } h \in S_R \text{ is a node-task.} \end{cases} \\
 & \forall k \in K, \forall S \subseteq C, (u,v) \in \gamma_R(S), h \in S_R
 \end{aligned}$$

The figure 2.2 illustrates the connectivity constraints. In the figure: $S = \{2, 3, 4\}$, $S_R = \{2, 4\}$, $\gamma_R(S) = \emptyset$, $E_R^+(S) = \{(2, 1)\}$, $E_R^-(S) = \{(1, 2)\}$, $A_R^+(S) = \emptyset$, $A_R^-(S) = \{(0, 3)\}$, $E^+(S) = \{(4, 5), (4, 0), (2, 0), (2, 1)\}$, $E^-(S) = \{(5, 4), (0, 4), (0, 2), (1, 2)\}$, $A^+(S) = \emptyset$, $A^-(S) = \{(0, 3)\}$, then the corresponding connectivity constraints are:

$$\begin{aligned}
 & x_{21}^k + x_{12}^k + x_{03}^k + y_{45}^k + y_{40}^k + y_{20}^k + y_{21}^k + y_{54}^k + y_{04}^k + y_{02}^k + y_{12}^k + y_{03}^k \geq 2z_2^k \quad \forall k \in K \\
 & x_{21}^k + x_{12}^k + x_{03}^k + y_{45}^k + y_{40}^k + y_{20}^k + y_{21}^k + y_{54}^k + y_{04}^k + y_{02}^k + y_{12}^k + y_{03}^k \geq 2z_4^k \quad \forall k \in K
 \end{aligned}$$

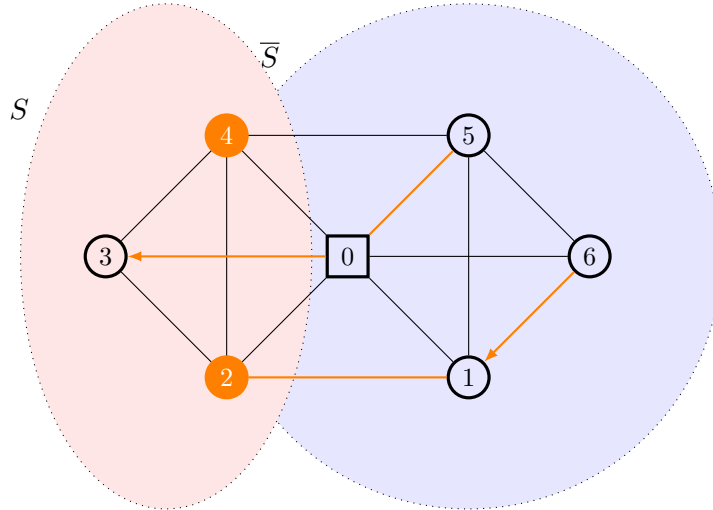


Figure 2.2: Connectivity constraints illustration

Finally, the following constraints define the variable domains.

2. PROBLEM DEFINITION

$$x_{ij}^k \in \{0, 1\}, \quad \forall k \in K, (i, j) \in A_R \cup E_R \quad (2.8)$$

$$y_{ij}^k \in \mathcal{Z}_+, \quad \forall k \in K, (i, j) \in A \cup E \quad (2.9)$$

$$z_i^k \in \{0, 1\}, \quad \forall k \in K, i \in C_R \quad (2.10)$$

The complete model is the following:

$$\begin{aligned} \text{Minimize } \lambda = & \sum_{k \in K} \sum_{(i,j) \in E_R} c_{ij}(x_{ij}^k + x_{ji}^k) + \sum_{k \in K} \sum_{(i,j) \in A_R} c_{ij}x_{ij}^k + \\ & \sum_{k \in K} \sum_{(i,j) \in E} c_{ij}(y_{ij}^k + y_{ji}^k) + \sum_{k \in K} \sum_{(i,j) \in A} c_{ij}y_{ij}^k \end{aligned} \quad (2.1)$$

$$\sum_{k \in K} (x_{ij}^k + x_{ji}^k) = 1, \forall (i, j) \in E_R \quad (2.2)$$

$$\sum_{k \in K} x_{ij}^k = 1, \forall (i, j) \in A_R \quad (2.3)$$

$$\sum_{k \in K} z_i^k = 1, \forall i \in C_R \quad (2.4)$$

$$\sum_{(i,j) \in E_R} d_{ij}(x_{ij}^k + x_{ji}^k) + \sum_{(i,j) \in A_R} d_{ij}x_{ij}^k + \sum_{i \in C_R} q_i z_i^k \leq Q, \forall k \in K \quad (2.5)$$

$$\begin{aligned} \sum_{j:(i,j) \in A_{Ri}^+} x_{ij}^k + \sum_{j:(i,j) \in A_i^+} y_{ij}^k - \sum_{j:(j,i) \in A_{Ri}^-} x_{ji}^k - \sum_{j:(j,i) \in A_i^-} y_{ji}^k = \\ \sum_{j:(j,i) \in E_{Ri}^-} x_{ji}^k + \sum_{j:(j,i) \in E_i^-} y_{ji}^k - \sum_{j:(i,j) \in E_{Ri}^+} x_{ij}^k - \sum_{j:(i,j) \in E_i^+} y_{ij}^k, \end{aligned} \quad (2.6)$$

$$\forall k \in K, i \in V$$

2.2 Mathematical formulation

$$\begin{aligned}
& \sum_{\forall (i,j) \in E_R^+(S)} x_{ij}^k + \sum_{\forall (j,i) \in E_R^-(S)} x_{ji}^k + \sum_{\forall (i,j) \in A_R^+(S)} x_{ij}^k + \sum_{\forall (j,i) \in A_R^-(S)} x_{ji}^k \quad (2.7) \\
& \sum_{\forall (i,j) \in E^+(S)} y_{ij}^k + \sum_{\forall (j,i) \in E^-(S)} y_{ji}^k + \sum_{\forall (i,j) \in A^+(S)} y_{ij}^k + \sum_{\forall (j,i) \in A^-(S)} y_{ji}^k \geq \\
& \quad \begin{cases} 2(x_{uv}^k + x_{vu}^k) & \text{if } (u,v) \in \gamma_R(S) \text{ is an edge-task;} \\ 2x_{uv}^k & \text{if } (u,v) \in \gamma_R(S) \text{ is an arc-task;} \\ 2z_h^k & \text{if } h \in S_R \text{ is a node-task.} \end{cases} \\
& \quad \forall k \in K, \forall S \subseteq C, (u,v) \in \gamma_R(S), h \in S_R
\end{aligned}$$

$$x_{ij}^k \in \{0, 1\}, \quad \forall k \in K, (i, j) \in A_R \cup E_R \quad (2.8)$$

$$y_{ij}^k \in \mathcal{Z}_+, \quad \forall k \in K, (i, j) \in A \cup E \quad (2.9)$$

$$z_i^k \in \{0, 1\}, \quad \forall k \in K, i \in C_R \quad (2.10)$$

2. PROBLEM DEFINITION

3

Valid inequalities and separation algorithms

3.1 Connectivity constraints separation

The constraints 2.7 are critical because they should be written for each subset of nodes $S \subseteq C$. The number of subsets of set C is $2^{|C|}$, this is a very big number and it makes computational intractable the real instances. We handle this problem by solving a relaxed formulation of 2.11 in which the constraints 2.7 are written only for the R-Sets. The solution of this relaxation is likely infeasible (because it contains subcycles), therefore we use the separation procedures described in the following to check the violations at each node of the search tree.

Connectivity constraints are initially checked by running a very fast heuristic algorithm and using an exact separation procedure whenever the heuristic fails.

3.1.1 Heuristic Algorithm

The heuristic algorithm is based on an idea of Fischetti, Salazar and Toth Fischetti et al. (1997). Given a solution of the relaxed MCGRP model, for a vehicle index k three vectors of variables may be defined: x^k , y^k , and z^k . Specifically, x^k is a $(|A_R| + 2|E_R|)$ -dimensional vector only including variables x_{ij}^k corresponding to k , y^k is a $(|A| + 2|E|)$ -dimensional vector only including variables y_{ij}^k corresponding to k , z^k is a $(|C_R|)$ -dimensional vector defined by variables z_i^k corresponding to k . The heuristic separation procedure is described in algorithm 1.

3. VALID INEQUALITIES AND SEPARATION ALGORITHMS

Algorithm 1: Heuristic separation for the connectivity inequalities

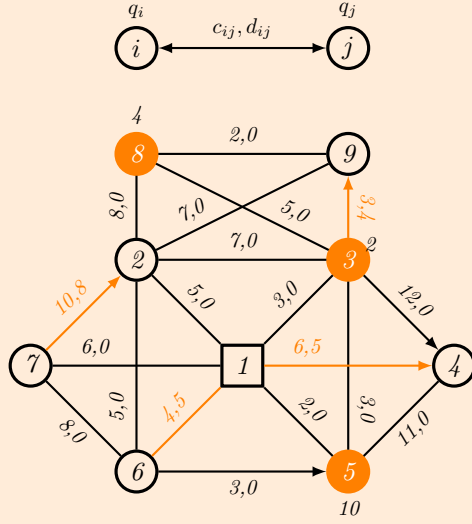
Require: The optimal solution of the relaxed formulation $(\bar{x}, \bar{y}, \bar{z})$;

Ensure: violated inequalities 2.7;

- 1: **for all** vehicle k **do**
 - 2: extract an undirected auxiliary graph G^k induced in G by all the links (i, j) such that $\bar{x}_{ij}^k > 0$ or $\bar{y}_{ij}^k > 0$;
 - 3: find the connected components of G^k and let p be the number of connected components not containing the depot and such that $V_1^k, V_2^k, \dots, V_p^k$ are the corresponding node sets;
 - 4: build a new graph \bar{G}^k with a node for each node set V_i^k , $i = 1, \dots, p$. Each pair of nodes (i, r) corresponding to V_s^k and V_r^k ($s, r = 1, \dots, p$, $s \neq r$) is connected by an edge whose cost is the sum of \bar{x}_{ij}^k and \bar{y}_{ij}^k for each $i \in V_s^k$ and $j \in V_r^k$ or $i \in V_r^k$ and $j \in V_s^k$. Obviously, if there is no link in G among the nodes belonging to V_s^k and V_r^k then the cost of the edge is zero.
 - 5: find the maximum spanning tree on the graph \bar{G}^k . At each step of the tree construction, a node is selected. Let V_s^k be the node set associated to the node s . If the connectivity constraint associated with the subset $S^k = V_s^k$ is violated, then it is introduced in the current relaxed formulation;
 - 6: once the spanning tree is complete, another check for violated constraints is performed by removing in turn the edges of the tree.
 - 7: **end for**
-

Example 1 *The following picture represents an MCGRP instance where $Q = 10$ and the cost and demand associated with each link are represented by couple (c_{ij}, d_{ij}) , while the demand of each required node is represented by (q_i) .*

3.1 Connectivity constraints separation



The optimal solution of the relaxed model 2.11 obtained by considering constraints 2.7 only defined for the R-Set is described as follows:

$$\begin{array}{llll}
 x_{72}^1 = 1 & y_{17}^1 = y_{21}^1 = 1 & r_1 = \{1 - 7 - 2 - 1\} & c_1 = 21 \\
 x_{39}^2 = 1 & y_{98}^2 = y_{83}^2 = 1 & z_3^2 = z_8^2 = 1 & r_2 = \{3 - 9 - 8 - 3\} & c_2 = 10 \\
 & y_{15}^3 = y_{51}^3 = 1 & z_5^1 = 1 & r_3 = \{1 - 5 - 1\} & c_3 = 4 \\
 x_{14}^4 = x_{16}^4 = 1 & y_{45}^4 = y_{51}^4 = y_{61}^4 = 1 & & r_4 = \{1 - 4 - 5 - 1 - 6 - 1\} & c_4 = 27
 \end{array}$$

The optimal value is 62. The connectivity constraints introduced into the initial formulation for $S = \{3, 9\}$ and $k = 2$ are satisfied by the current solution. Nevertheless such solution is infeasible for the MCGRP since it violates the connectivity constraint defined by subset $S = \{3, 8, 9\}$:

$$y_{31}^2 + y_{32}^2 + y_{34}^2 + y_{35}^2 + y_{82}^2 + y_{92}^2 + y_{13}^2 + y_{23}^2 + y_{53}^2 + y_{28}^2 + y_{29}^2 \geq 2x_{39}^2$$

The graph G^2 generated at step 4 of the heuristic separation algorithm contains only a node associated with the connected component defined by R-set $V_1^2 = \{3, 8, 9\}$.

3. VALID INEQUALITIES AND SEPARATION ALGORITHMS

3.1.2 Exact Algorithm

The exact separation algorithm follows the outline provided by Benavent et al. (2000) for the Capacitated ARP. Specifically, for each vehicle index k , let $G^k(w)$ be an undirected graph including the depot and induced by the edges $(i, j) \in E$ with a capacity defined by $w_{ij}^k = \bar{x}_{ij}^k + \bar{x}_{ji}^k + \bar{y}_{ij}^k + \bar{y}_{ji}^k > 0$ and the edges corresponding to the arcs $(i, j) \in A$ with a capacity defined by $w_{ij}^k = \bar{x}_{ij}^k + \bar{y}_{ij}^k > 0$. Constraints (2.7) can be separated in polynomial time by solving a min-cut separating the depot from each node of $G^k(w)$ (algorithm 2).

Algorithm 2: Exact connectivity inequalities separation

Require: The optimal solution of the relaxed formulation $(\bar{x}, \bar{y}, \bar{z})$;

Ensure: violated inequalities 2.7;

- 1: **for all** vehicle k **do**
- 2: extract an auxiliary directed graph G^k induced in G by all the links $(i, j) \in E \cup A$. Each arc $(i, j) \in A$ corresponds to an arc (i, j) in G^k with weight $w_{ij} = \bar{x}_{ij}^k + \bar{y}_{ij}^k$; while each edge $(i, j) \in E$ is splitted into a couple of arcs in G^k : (i, j) with weight $w_{ij} = \bar{x}_{ij}^k + \bar{y}_{ij}^k$ and (j, i) with weight $w_{ji} = \bar{x}_{ji}^k + \bar{y}_{ji}^k$;
- 3: **for all** $(i, j) \in E_R : x_{ij}^k + x_{ji}^k > 0$ **do**
- 4: compute the min-cuts separating the depot node and i and the depot node and j . Let $\delta(S_i)$ and $\delta(S_j)$ be the min-cuts and F_i and F_j the capacities of these cuts.
- 5: **if** $\max\{F_i, F_j\} < 2(x_{ij}^k + x_{ji}^k)$ **then**
- 6: a violated connectivity constraint has been found.
- 7: **end if**
- 8: **end for**
- 9: **for all** $(i, j) \in A_R | x_{ij}^k > 0$ **do**
- 10: compute the min-cuts separating the depot node and i and the depot node and j . Let $\delta(S_i)$ and $\delta(S_j)$ be the min-cuts and F_i and F_j the capacities of these cuts.
- 11: **if** $\max\{F_i, F_j\} < 2x_{ij}^k$ **then**
- 12: a violated connectivity constraint has been found.
- 13: **end if**

```

14:  end for
15:  for all  $i \in V_R | z_i^k > 0$  do
16:      compute the min-cut separating the depot node and  $i$ . Let  $\delta(S_i)$  be the min-cut
          and  $F_i$  the capacity of this cut.
17:      if  $F_i < 2z_i^k$  then
18:          a violated connectivity constraint has been found.
19:      end if
20:  end for
21: end for

```

3.2 Surrogate inequalities

Optimally solving the capacitated routing problems is a hard issue to be addressed due to the capacity constraints and the large number of vehicles to be used. Belenguer and Benavent (1998) and Belenguer and Benavent (2003) observed a considerable improvement in a surrogate lower-bounding model for the capacitated ARP obtained by aggregating the three-index edge variables with respect to the indices of the vehicles. The authors proposed an integer programming model aimed at minimizing the total deadheading cost, by ensuring no violation of aggregate capacity constraints and aggregate odd edge cutset constraints. They also defined new classes of disjoint path inequalities to tighten the lower bounds obtained through a cutting plane algorithm. The computational results presented in their papers show that the capacity and odd edge cutset constraints contribute more than the disjoint path inequalities to improve the bounds. In a large number of benchmark instances their lower bounds prove the optimality of the best known upper bounds. As following these considerations, an initial subset of capacity and odd edge cutset inequalities have been introduced into the relaxed MCGRP formulation, with the aim of improving the lower bound at the root node of the search tree.

3. VALID INEQUALITIES AND SEPARATION ALGORITHMS

3.2.1 Capacity and odd-edge inequalities

Let $G^w = (V^w, E^w)$ a windy graph obtained from G by replacing each arc (i, j) with an edge. This transformation helps to make our problem closer to the original one by Belenguer and Benavent. The cost on a new edge of G^w is the same of the cost of the correspondent arc (i, j) in the original graph if the edge is used from i to j , it is ∞ otherwise. Note that $V^w = V$ and $C^w = V^w \setminus \{1\} = C$. The notation introduced in the Section 2.1 can be transposed to G^w . Therefore, E_R^w denotes the set of edge-tasks, C_R^w denote the set of node-tasks, $E^w(S)$ and $E_R^w(S)$ denote respectively the (cut)sets of edges and edge-tasks with one endpoint in S and the other out of S , S_R^w denotes the set of node-tasks in S and $\gamma_R^w(S)$ denotes the set of edge-tasks with both endpoints in S .

Let θ_{ij} be the total number of times that $(i, j) \in E^w$ is deadheaded by the vehicles. We say that a given vehicle *cross* a cutset $E^w(S)$ whenever it traverses an edge $(i, j) \in E^w(S)$. A group of valid inequalities (*capacity constraints*) can be expressed in terms of the aggregated variables θ_{ij} :

$$\sum_{(i,j) \in E^w(S)} \theta_{ij} \geq 2\eta(S) - |E_R^w(S)| \quad \forall S \subseteq C^w, \quad (3.1)$$

where $\eta(S) = \lceil (\sum_{(i,j) \in \gamma_R^w(S) \cup E_R^w(S)} d_{ij} + \sum_{i \in S_R^w} q_i) \setminus Q \rceil = \lceil D(S) \setminus Q \rceil$ represents the minimum number of vehicles needed to service all the edge-tasks in the cutset $E_R^w(S)$ and inside S . In fact, a vehicle which services some edges in $\gamma_R^w(S) \cup E_R^w(S)$ and/or some nodes in S_R^w crosses $E^w(S)$ at least twice, so the number of deadheading edges used by such vehicle to cross the cutset $E^w(S)$ is at least $2\eta(S) - |E_R^w(S)|$. Capacity constraints provide the link between the packing and routing structure of the MCGRP polyhedron.

The resulting graph associated to the feasible solutions of MCGRP must be an even graph, i.e., all its nodes must have an even degree. It can be easily shown that a cutset must contain an even number of edges. Therefore, for each cutset containing an odd number of edge-tasks, at least one edge in the cut must be deadheaded. This fact is expressed by the so-called *odd edge cutset constraints*:

$$\sum_{(i,j) \in E^w(S)} \theta_{ij} \geq 1 \quad \forall S \subseteq C^w, \quad \text{with } |E_R^w(S)| \text{ odd.} \quad (3.2)$$

The constraints (3.1) and (3.2) can be rewritten in the following unified way:

$$\sum_{(i,j) \in E^w(S)} \theta_{ij} \geq \alpha(S) \quad \forall S \subseteq C^w, \quad (3.3)$$

where

$$\alpha(S) = \begin{cases} \max \{2\eta(S) - |E_R^w(S)|, 1\} & \text{if } |E_R^w(S)| \text{ is odd,} \\ \max \{2\eta(S) - |E_R^w(S)|, 0\} & \text{if } |E_R^w(S)| \text{ is even.} \end{cases}$$

In order to express (3.3) in terms of the original variables of the MCGRP model, we observe that

$$\theta_{ij} = \begin{cases} \sum_{k \in K} (y_{ij}^k + y_{ji}^k) & \text{if } (i, j) \text{ is an edge in } G, \text{ i.e., } c_{ij} = c_{ji} \text{ in } G^w, \\ \sum_{k \in K} y_{ij}^k & \text{if } (i, j) \text{ is an arc in } G, \text{ i.e., } c_{ji} = \infty \text{ in } G^w. \end{cases}$$

The sets in the formulas (e.g., $E_R^w(S)$) can be easily transformed with respect to the original graph G .

In the following we extend to our problem the relevant identification procedures defined by Belenguer and Benavent (2003) for the Capacitated ARP. We denote by $\bar{\theta}_{ij}$ the current optimal value for the aggregated variable θ_{ij} .

3.2.2 Capacity inequalities separation

The separation problem related to the capacity constraints is quite difficult to optimally solve, mainly because the right side of (3.1) involves the integer up rounding of $\frac{D(S)}{Q}$. The following procedure is devoted to separate a fractional relaxation of the capacity constraints in polynomial time (see Augerat et al. (1998)). In our case, the fractional capacity constraints are formulated as follows:

$$\sum_{(i,j) \in E^w(S)} \theta_{ij} \geq 2(D(S) \setminus Q) - |E_R^w(S)| \quad \forall S \subseteq C^w. \quad (3.4)$$

Since $\eta(S) \geq D(S) \setminus Q$, the inequalities (3.1) dominate the inequalities (3.4). The fractional capacity constraints can be effectively identified by using a procedure similar to the one described by Belenguer and Benavent (2003). It consists in solving a maximum flow problem on a graph \bar{G}^w obtained from G^w by adding an artificial node σ and edges connecting σ to the other nodes in G^w . The capacity of each edge (i, j) in \bar{G}^w is

3. VALID INEQUALITIES AND SEPARATION ALGORITHMS

denoted by b_{ij} and defined as follows:

$$b_{ij} = \begin{cases} \bar{\theta}_{ij} & \text{if } (i, j) \in E^w \setminus E_R^w, \\ \bar{\theta}_{ij} + 1 - \frac{d_{ij}}{Q} & \text{if } (i, j) \in E_R^w, \\ \frac{2}{Q}q_i + \frac{1}{Q} \sum_{(i,h) \in E_R^w(i)} d_{ih} & \text{if } i \in V^w \text{ and } j = \sigma. \end{cases}$$

Let v be the minimum capacity of the cut defined by $S \cup \{\sigma\}$ and obtained by solving a maximum flow problem on \bar{G}^w between nodes 1 and σ . Observe that S represents the set of the original nodes defining this optimal cut. Let us define P as $\frac{2}{Q}(\sum_{(i,j) \in E_R^w} d_{ij} + \sum_{i \in V_R^w} q_i)$. The slack of constraint (3.4) with respect to S is obtained by subtracting P to v . In fact:

$$\begin{aligned} v - P &= \\ &= \sum_{(i,j) \in E_R^w(S)} b_{ij} + \sum_{(i,j) \in E^w(S) \setminus E_R^w(S)} b_{ij} + \sum_{i \in V^w \setminus S} b_{i\sigma} - \frac{2}{Q} \sum_{(i,j) \in E_R^w} d_{ij} - \frac{2}{Q} \sum_{i \in V_R^w} q_i \\ &= \sum_{(i,j) \in E_R^w(S)} \bar{\theta}_{ij} + |E_R^w(S)| - \frac{1}{Q} \sum_{(i,j) \in E_R^w(S)} d_{ij} + \sum_{(i,j) \in E^w(S) \setminus E_R^w(S)} \bar{\theta}_{ij} + \frac{2}{Q} \sum_{i \in V^w \setminus S} q_i \\ &\quad + \frac{1}{Q} \sum_{i \in V^w \setminus S} \sum_{(i,h) \in E_R^w(i)} d_{ih} - \frac{2}{Q} \sum_{(i,j) \in E_R^w} d_{ij} - \frac{2}{Q} \sum_{i \in V_R^w} q_i \\ &= \sum_{(i,j) \in E^w(S)} \bar{\theta}_{ij} + |E_R^w(S)| + \frac{2}{Q} \sum_{(i,j) \in \gamma_R^w(V^w \setminus S)} d_{ij} + \frac{2}{Q} \sum_{i \in V^w \setminus S} q_i \\ &\quad - \frac{2}{Q} \sum_{(i,j) \in E_R^w} d_{ij} - \frac{2}{Q} \sum_{i \in V_R^w} q_i \\ &= \sum_{(i,j) \in E^w(S)} \bar{\theta}_{ij} + |E_R^w(S)| - \frac{2}{Q} \sum_{(i,j) \in \gamma_R^w(S) \cup E_R^w(S)} d_{ij} - \frac{2}{Q} \sum_{i \in S_R^w} q_i \\ &= \sum_{(i,j) \in E^w(S)} \bar{\theta}_{ij} + |E_R^w(S)| - 2 \frac{D(S)}{Q}. \end{aligned}$$

Therefore, if $v - P < 0$ then the constraints (3.4) and (3.1) are violated for S . If $v - P \geq 0$ then no constraint of type (3.4) is violated, but (3.1) could be violated for S . Hence, this procedure is used to generate a set S for which (3.1) are checked for possible violations (see algorithm 3). Whenever the fractional capacity identification procedure fails, then a different method is used to identify violated capacity constraints of type (3.1). Such method is based on a demand perturbation. Observe that the slack of (3.4) can be approximated to the slack of (3.1) computing the first one in a graph $\bar{G}^w(p)$

3.2 Surrogate inequalities

where the demands of edge-tasks and node-tasks are multiplied for $(1 + p)$, $0 < p < 1$. Then, the fractional capacity identification procedure based on the minimum capacity cut problem is applied again on $\bar{G}^w(p)$ (see algorithm 4).

Algorithm 3: Capacity inequalities separation

Require: The graph $G^w = (V^w, E^w)$ and a solution $\bar{\theta}^{|E^w|}$;

Ensure: violated inequalities 3.1;

- 1: generate the support graph $G(\bar{\theta})$;
 - 2: calculate the maximum flow from node 1 to $n - 1$;
 - 3: **if** $\phi(S \cup n + 1)^* - \frac{2}{Q} \sum_{(i,j) \in E_R^w} d_{ij} - \frac{2}{Q} \sum_{i \in V^w} \bar{q}_i < 0$ **then**
 - 4: add violated constraint 3.1 for $S = S^*$;
 - 5: **else**
 - 6: call algorithm 4;
 - 7: **end if**
-

Algorithm 4: Perturbed demand procedure

Require: The graph $G^w = (V^w, E^w)$;

Ensure: violated inequalities 3.1;

- 1: Set $N_p = 0$;
 - 2: **while** $N_p < 10$ **do**
 - 3: choose p randomly in the interval $(0, 1)$;
 - 4: $N_p = N_p + 1$;
 - 5: $\forall (i, j) \in E_R^w d'_{ij} = d_{ij} * (1 + p)$;
 - 6: $\forall i \in V_R^w q'_i = q_i * (1 + p)$;
 - 7: apply the algorithm 3 with the new demands;
 - 8: **end while**
-

3.2.3 Odd-edge inequalities separation

The odd edge cutset inequalities can be exactly separated in polynomial time. We implemented both an exact algorithm and a faster heuristic procedure. The former is

3. VALID INEQUALITIES AND SEPARATION ALGORITHMS

inspired by an algorithm proposed by Padberg and Rao (1982), whereas the heuristic procedure is inspired by an algorithm described in Corberán et al. (2001).

The inequalities to be identified are the following:

$$\sum_{(i,j) \in E^w(S)} \theta_{ij} \geq 1 \quad \forall S \subseteq C^w, \quad \text{with } |E_R^w(S)| \text{ odd.} \quad (3.2)$$

The exact separation procedure described in Padberg and Rao (1982) is referred to the *b-matching* problem. Such problem can be formulated as follows:

$$\text{Maximize } c^T x \quad (3.5a)$$

$$Ax \leq b \quad (3.5b)$$

$$x \in \mathcal{Z}_+^m \quad (3.5c)$$

where c is a m -vector of non-negative coefficients, $A \in \{0, 1\}^{n \times m}$ is the incidence matrix of an undirected graph with n nodes and m edges and b is a vector of n positive integer numbers, each representing the maximum number of edges incident to each node of the graph. In the problem 3.5 not all the basic feasible solutions are automatically integer (the graph is not bipartite). The valid inequalities that allow to obtain integer basic feasible solutions are called *matching constraints* or *blossom inequalities*.

Let $G = (V, E)$ be an undirected graph with incidence matrix A and let $W \subseteq V$ a subset of nodes of G (or a subset of rows of A) such that $b(W) = \sum_{i \in W} b_i$ is odd. The inequality:

$$x(W) = \sum_{e \in \gamma(W)} x_e \leq \frac{1}{2}(b(W) - 1) \quad (3.6)$$

is valid for 3.5, where $\gamma(W)$ is the set of edges with both endpoints inside W . Let s be the vector of slack variables of the inequalities $Ax \leq b$ and $s(W) = \sum_{i \in W} s_i$. Since $Ax + s = b$, for the subset of rows from the matrix A corresponding to the vertices in W , then we can write $2x(W) + x(E(W)) + s(W) = b(W)$, where $x(E(W)) = \sum_{e \in E(W)} x_e$ and $E(W)$ is the edge cutset defined by W . Dividing by 2, we have: $x(W) = \frac{1}{2}[b(W) - (x(E(W)) + s(W))]$. The inequality 3.6 is valid if and only if:

$$x(E(W)) + s(W) \geq 1, \forall W \subseteq V, \text{ such that } b(W) \text{ is odd} \quad (3.7)$$

In the particular case $s(W) = 0$ (all the constraints of 3.5 corresponding to the vertices of W satisfied as equalities), condition 3.7 becomes:

$$x(E(W)) \geq 1, \forall W \subseteq V, b(W) \text{ odd} \quad (3.8)$$

Setting $\theta = x$ and b_i the number of edges-tasks incident to each node of $S = W$, since $b(S) = 2|\gamma_R(S)| + |E_R(S)|$, $b(S)$ is odd if and only if $|E_R(S)|$ is odd, then:

$$\theta(E(S)) \geq 1, \forall S \subseteq V, |E_R(S)| \text{ odd} \quad (3.9)$$

Constraints 3.2 are formally equal to 3.8, then the separation of 3.2 may be performed by using the same exact separation algorithm of 3.8 whenever $s(W) = 0$.

Let $\bar{\theta} \in \mathcal{Z}_+^{|E^w|}$ the solution vector at the current iteration of the cutting plane algorithm and let $G(\bar{\theta})$ the graph induced by the edges of E^w with $\bar{\theta}_{ij} > 0$. Each edge of $G(\bar{\theta})$ has a capacity $\bar{\theta}_{ij} > 0$. Let T be the set of odd nodes of $G(\bar{\theta})$ (the nodes with an odd number of incident edge-tasks in the original graph G). We construct the *Cut-Tree* by using the Gomory-Hu algorithm (algorithm 5).

Algorithm 5: Cut-tree

Require: A graph $G(\bar{\theta}) = (\bar{V}, \bar{E})$ and a set $T \subset \bar{V}$ of terminal vertices;

Ensure: A Cut-tree \mathcal{C} ;

- 1: $\mathcal{L} := \bar{V}$
- 2: **while** $T \neq \bar{V}$ **do**
- 3: Select a terminal vertex t from T and let $R \in \mathcal{L}$ be the supernode of \mathcal{C} that contains t . Let r be the representative vertex of R ;
- 4: Let $G_R(\bar{\theta})$ the induced graph obtained replacing all the supernodes S_1, \dots, S_l of the cut-tree \mathcal{C} and incident to R , with the vertices $s_i, i = 1, \dots, l$;
- 5: Let $\lambda_{G_R(\bar{\theta})}(r, t)$ the maximum flow from the source node r to the destination node t computed on the reduced graph $G_R(\bar{\theta})$ and let $\delta(X)$ the minimum (r, t) -cut in $G_R(\bar{\theta})$. If $G_R(\bar{\theta})$ is not connected is not possible to send a flow from r to t , then the maximum flow is 0 and $\delta(X) = (\bar{V}_{C_r}, \bar{V}_{C_t})$ where \bar{V}_{C_r} is the set of vertices of the connected component the contains r , and \bar{V}_{C_t} is the set of vertices of the connected component that contains t ;

3. VALID INEQUALITIES AND SEPARATION ALGORITHMS

- 6: Let $\mathcal{L} = (\mathcal{L} \setminus \{R\}) \cup (\{R \cap X\} \cup (\{R \cap \bar{X}\}))$. The supernode R is replaced by the supernodes $R \cap X$ and $R \cap \bar{X}$ connected by an edge which weight is $f(R \cap X, R \cap \bar{X}) = \lambda_{G_R(\bar{\theta})}(r, t)$
 - 7: For each $i = 1, \dots, l$ replace each edge (R, S_i) of the cut-tree \mathcal{C} with a new edge $(R \cap X, S_i)$ with weight $f(R \cap X, S_i) = f(R, S_i)$ if $s_i \in X$ or with a new edge $(R \cap \bar{X}, S_i)$ with weight $f(R \cap \bar{X}, S_i) = f(R, S_i)$ if $s_i \in \bar{X}$;
 - 8: **if** $R \cap X$ or $R \cap \bar{X}$ contains only the terminal vertex t **then**
 - 9: $T = T \setminus \{t\}$
 - 10: **end if**
 - 11: **end while**
-

In general, let $G = (V, E, \gamma)$ a undirected weighted graph, with weights $\gamma \in \mathcal{Q}_+^{|E|}$ on the edges. Let $T \subset V$ a subset made up by an even number of odd vertices. A cut $\delta(U)$ is defined T -odd, or simply *odd*, if $|T \cap U|$ is an odd number. The minimum odd cut problem consists in finding an odd cut $\delta(U)$ with the minimum weight $\gamma(\delta(U))$ (algorithm 6).

Algorithm 6: Exact Odd Edge cut separation

Require: The optimal solution $\bar{\theta} \in \mathcal{Z}_+^{|E^w|}$ at any iteration of the cutting plane algorithm;

Ensure: A minimum odd set S for which constraints 3.2 are violated;

- 1: Let $\epsilon = 1$;
- 2: Let $S = \emptyset$;
- 3: Generate the graph $G(\bar{\theta}) = (\bar{V}, \bar{E})$ induced in G by the current solution: $G(\bar{\theta}) = \text{Graph}(G, \bar{\theta})$;
- 4: Find the set $T \subseteq \bar{V}$ of the terminal vertices (odd vertices in G): $T = \text{GetOdd}(G)$;
- 5: Invoke algorithm 5 on $G(\bar{\theta})$ with the set of terminal vertices T and obtain the cut-tree $\mathcal{C}_{G(\bar{\theta})}$: $\mathcal{C}_{G(\bar{\theta})} = \text{CutTree}(G(\bar{\theta}), T)$;
- 6: **for all** $|T| - 1$ edges of the cut-tree $\mathcal{C}_{G(\bar{\theta})}$ **do**
- 7: Let $\delta(U^e)$ be the cutset induced by the edge $e = (S_i, S_j)$, where S_i and S_j are two supernodes of the cut-tree. Note that U^e is a set of supernodes of the cut-tree;
- 8: checking for any violation generated by the cut: if $|T \cap U^e|$ is odd and $\bar{w}(\delta(U^e)) = f(S_i, S_j) < \epsilon$, store in $S = S \cup \{S^e = \text{GetVertices}(G, U^e)\}$ the vertices of

3.2 Surrogate inequalities

the original graph G included yet into the supernodes of U^e , and such that 3.2 is violated. Note that $f(S_i, S_j)$ represents the maximum flow on the edge $e = (S_i, S_j)$ of the cut-tree.

9: **end for**

3. VALID INEQUALITIES AND SEPARATION ALGORITHMS

4

An Upper bound

An initial feasible *MCGRP* solution is generated by running a heuristic algorithm based on a simple construction phase, followed by an improvement phase that implements a local-search procedure.

Two different procedures are used with the aim of constructing an initial feasible solution: the *partition-first-route-next* method proposed by Gutiérrez et al. (2002) and a *clustering* procedure followed by the solution of a *GRP* problem for each cluster. Then, an iterative local search procedure is performed to improve the solution obtained in the previous phase according to the pseudo-code of the overall algorithm in 7.

Algorithm 7: Heuristic

Require: An *MCGRP* instance;

Ensure: A feasible solution $\mathcal{S} = (x, y, z)$;

1: $\mathcal{S}' = \text{InitialSolution}()$;

2: $\mathcal{S} = \text{LocalSearch}(\mathcal{S}')$;

4.1 The initial solution

An initial solution is obtained by using the heuristic procedure of Gutiérrez et al. (2002). Whenever the solution is m -infeasible (it does not contain m routes), then a clustering method is used (see algorithm 8).

Algorithm 8: Initial solution

4. AN UPPER BOUND

Require: An *MCGRP* instance;

Ensure: A feasible solution $\mathcal{S} = (x, y, z)$;

- 1: $\mathcal{S} = \text{Partition} - \text{First} - \text{Route} - \text{Next}()$;
 - 2: **if** $\text{Routes}(\mathcal{S}) \neq |K|$ **then**
 - 3: $\mathcal{S} = \text{Clustering}()$;
 - 4: **end if**
-

A detailed description of how the two procedures work is reported in the following section.

4.1.1 Partition-First-Route-Next

This heuristic procedure constructs one route at a time. Firstly, for each required arc, edge or isolated vertex it finds a minimum cost initial route connecting the required element to the depot. Then the remaining largest initial route is selected and other required elements (edge, arc or node) are serviced at the minimum cost, according to the following preferences. The required elements nearest to the one defining the initial route, and according to the capacity constraint are firstly considered. Then, all the deadhead elements which demands do not exceed the residual capacity of the vehicle are taken into account for possible insertion. Finally, all the required elements very close to the route and with a demand not greater than $0.9Q$ are considered.

More notation is required in order to understand the outlines referred to the described procedures.

- the depot is identified by the vertex 1;
- Q_t : total demand in the graph;
- C_a : accumulated cost of the assigned routes;
- D_a : accumulated demand of the assigned routes;
- *required element*: each required isolated vertex (non incident with required arcs or edges) and each required arc or edge together with its two incident vertices. A required element is denoted by (i, j) . If $i = j$ this element is a (isolated) vertex;

- B : the set of required elements not yet inserted in a route;
- $R(i, j)$: a route containing the depot and the required element (i, j) ;
- $C(i, j)$: total cost of the route $R(i, j)$;
- $D(i, j)$: total demand of the route $R(i, j)$;
- N : number of assigned routes;
- $SP(u, v)$: shortest path from vertex u to vertex v ;
- $CSP(u, v)$: cost of $SP(u, v)$ ($CSP(u, u) = 0$);
- q_0 : minimal demand corresponding to the required vertices (all), arcs and edges not yet assigned to any route;
- DF : demand needed to saturate the capacity Q of a route;
- M_0 : statistical median of the costs of all edges and arcs of the graph;
- $KI(u, v)$: insertion path from vertex u to vertex v . If $u \neq v$, $KI(u, v)$ is a segment of route between vertices corresponding to two consecutive required elements, including the depot inserted in the route, and such that this segment does not contain any other required element inserted in the route. Necessarily $KI(u, v) = SP(u, v)$. If $u = v$, $KI(u, u) = u$.

Algorithm 9: Partition-First-Route-Next

Require: An *MCGRP* instance;

Ensure: A feasible solution $\mathcal{S} = (x, y, z)$;

1: {Initialization}

2: Find the initial route for each $(i, j) \in B$:

- for each edge $(i, j) \in B$ its initial route is the one of least cost between $R(i, j) = SP(1, i) \cup (i, j) \cup SP(j, 1)$ and $R(j, i) = SP(1, j) \cup (j, i) \cup SP(i, 1)$;
- for each arc $(i, j) \in B$ its initial route is $R(i, j) = SP(1, i) \cup (i, j) \cup SP(j, 1)$;

4. AN UPPER BOUND

- for each isolated vertex $(i, i) \in B$ ($i = (i, i)$) its initial route is $R(i, i) = SP(1, i) \cup SP(i, 1)$.

3: For each pair $(u_1, u_2), (v_1, v_2)$ of required elements of the graph, including the depot, calculate the distance between them, defined as

$$\min\{CSP(u_1, v_1), CSP(u_1, v_2), CSP(u_2, v_1), CSP(u_2, v_2), \\ CSP(v_1, u_1), CSP(v_1, u_2), CSP(v_2, u_1), CSP(v_2, u_2)\}$$

4: $DA = 0, CA = 0, N = 0$

5: $N = N + 1, I = 0$

6: Select the initial route $R(i, j)$ with largest cost corresponding to a required element $(i, j) \in B$. Let $R(i, j) = SP(1, i) \cup (i, j) \cup SP(j, 1)$ be this route, consider the insertion paths: $KI(1, i), KI(j, 1), KI(1, 1), KI(i, i), KI(j, j)$, being (i, j) the first inserted required element in route N .

7: $D(i, j) = q_i + q_{ij} + q_j, DF = QD(i, j)$.

8: Put in order the demands of the remainder elements of B according to the instance of the required element to (i, j) in increasing order of distance. Given this order, calculate the maximal accumulated demand F_s (s implies that F_s corresponds to the first s elements) such that $F_s \leq DF$.

9: Let A_s be the set containing the first s required elements. If $A_s = \emptyset$ ($s = 0$) GO TO step 20

10: $D(i, j) = D(i, j) + F_s, DF = Q - D(i, j)$.

11: For each element of A_s not inserted yet in the route N , calculate the cost increment due to its insertion in each one of the new insertion paths $KI(u_i, v_i)$. This cost is defined in the following way:

- For each non inserted edge $(u, v) \in A_s, \min\{CSP(u_i, u) + c_{uv} + CSP(v, v_i) - CSP(u_i, v_i), CSP(u_i, v) + c_{uv} + CSP(u, v_i) - CSP(u_i, v_i)\};$
- For each non inserted arc $(u, v) \in A_s, CSP(u_i, u) + c_{uv} + CSP(v, v_i) - CSP(u_i, v_i);$

- For each non inserted isolated vertex $u \in A_s$, $CSP(u_i, u) + CSP(u, v_i) - CSP(u_i, v_i)$.
- 12: Select the insertion path $KI(u_i, v_i)$ and the required element $(u, v) \in A_s$ for which the minimum cost increment is reached in Step 11.
- Insert this required element in the route N by replacing $KI(u_i, v_i)$ with $SP(u_i, u) \cup (u, v) \cup SP(v, v_i)$. $R(i, j)$ is then updated: $R(i, j) = \{R(i, j) - KI(u_i, v_i)\} \cup SP(u_i, u) \cup (u, v) \cup SP(v, v_i)$. Remove $KI(u_i, v_i)$ from the list of insertion paths and add to this list the new insertion paths: $KI(u_i, u)$, $KI(v, v_i)$, $KI(u, u)$ and $KI(v, v)$ (note that occasionally $u_i = v_i$ and/or $u = v$).
- 13: $I = I + 1$
- 14: **if** $I < s$ **then**
- 15: GO TO step 11
- 16: **end if**
- 17: **if** $DF < q_0$ **then**
- 18: GO TO step 54
- 19: **end if**
- 20: Consider the demands not yet inserted in any route and located at vertices and links traversed by the route $R(i, j)$. Put in order these demands according to their distance to the depot in the route, going to or coming from the depot, in increasing order of distance. To do this, consider that a demand corresponding to an arc or edge is located in the center of the link. Given this order, calculate the maximal accumulated demand F_m (m implies that F_m corresponds to the first m elements) such that $F_m \leq DF$ and let F_n be the total non inserted demand located in $R(i, j)$.
- 21: **if** $F_m = 0$ ($m = 0$) **then**
- 22: GO TO step 32
- 23: **end if**
- 24: $D(i, j) = D(i, j) + F_m$
- 25: $DF = DF - F_m$
- 26: **if** $DF < q_0$ **then**
- 27: GO TO step 54
- 28: **end if**

4. AN UPPER BOUND

- 29: **if** $F_m = F_n$ **then**
- 30: GO TO step 41
- 31: **end if**
- 32: Put in increasing order the remaining demands not yet inserted in any route and located at vertices and links traversed by the route $R(i, j)$. Given this order, calculate the maximal accumulated demand F_p such that $F_p \leq DF$.
- 33: **if** $F_p = 0$ **then**
- 34: GO TO step 41
- 35: **end if**
- 36: $D(i, j) = D(i, j) + F_p$
- 37: $DF = DF - F_p$
- 38: **if** $DF < q_0$ OR $DF \leq 0.1Q$ **then**
- 39: GO TO step 54
- 40: **end if**
- 41: From among the required elements $(u, v) \in B$ not present in $R(i, j)$, not labeled and such that at least one of its demands q_u, q_{uv}, q_v , is different from zero and little or equal than DF , select the one that minimizes $\min\{CSP(u_i, u) + CSP(v, u_i) : u_i \text{ is a vertex belonging to a required element inserted in } R(i, j), \text{ including the depot}\}$ if this minimum is little or equal than $2M_0$.
- 42: **if** such required element (u, v) does not exist **then**
- 43: GO TO step 54
- 44: **end if**
- 45: Let $d_{uv} = q_u + q_{uv} + q_v$ be the demand of (u, v) .
- 46: **if** $d_{uv} \leq DF$ **then**
- 47: $d_0 = d_{uv}$ and GO TO step 50.
- 48: **end if**
- 49: From among the six elements $q_u, q_v, q_{uv}, q_u + q_v, q_u + q_{uv}, q_v + q_{uv}$, select the largest one little or equal than DF . Let d_0 be this element.
- 50: $D(i, j) = D(i, j) + d_0, DF = DF - d_0$ and:
- $R(i, j) = R(i, j) \cup SP(u_i, u) \cup (u, v) \cup SP(v, u_i)$ if d_0 contains q_{uv} .
 - $R(i, j) = R(i, j) \cup SP(u_i, u) \cup SP(u, u_i)$ if $d_0 = q_u$.
 - $R(i, j) = R(i, j) \cup SP(u_i, v) \cup SP(v, u_i)$ if $d_0 = q_v$.

- $R(i, j) = R(i, j) \cup SP(u_i, u) \cup (u, v) \cup SP(v, u_i)$ if $d_0 = q_u + q_v$ and $CSP(u_i, u) + c_{uv} + CSP(v, u_i) \leq CSP(u_i, u) + CSP(u, u_i) + CSP(u_i, v) + CSP(v, u_i)$.
 - $R(i, j) = R(i, j) \cup SP(u_i, u) \cup SP(u, u_i) \cup SP(u_i, v) \cup SP(v, u_i)$ otherwise.
- 51: **if** $DF > 0.10Q$ and $DF \geq q_0$ **then**
52: label (u, v) and GO TO step 41.
53: **end if**
54: Consider route $R(i, j)$ as a definitive route.
55: $DA = DA + D(i, j)$
56: $CA = CA + C(i, j)$
57: **if** $DA = Q$ **then**
58: STOP
59: **end if**
60: Update B , update the demands of its elements and for each required vertex become isolated, find its initial route and calculate its distances to the remaining required elements. Remove the label to the required elements labeled in step 50.
61: GO TO step 5.
-

The above heuristic has polynomial complexity. In fact, the steps with largest number of operations are: step 1 with complexity $O(|V|^3)$ (in the worst case, the set of its operations is dominated by the computation of the shortest path between each ordered pair of vertices in G , which are stored in such a way that we do not need to compute them in the following steps); step 8 with complexity $O(r^2)$ being $r = |A_R \cup E_R| + |V_R|$ (in the worst case, the set of its operations is dominated by the ordering of the demands of all required elements in G); step 11 that, after all the times that the heuristic passes through it, has complexity $O(r|V|^2)$ ($r|V|^2$ is an upper bound on the number of combinations between a required element and an insertion path) and finally step 41 whose complexity is $O(r|V|)$. Then, we may conclude that our heuristic has polynomial complexity upper-bounded by $O(s^3)$ being $s = \max\{|V|, r\}$ (note that $k \leq r$).

4.1.2 Clustering

The cluster's elements are the arcs, edges and vertices tasks. Each required arc, edge and vertex must be contained in exactly a cluster. The total demand of the elements

4. AN UPPER BOUND

in a cluster must not exceed Q (the capacity of the vehicle). The clustering algorithm proposed here starts with random “seeds” (a seed is a required element). The elements of the cluster C_j are the elements served by the j -th route. Consequently, if X is the set of clusters, then $|X| = |K|$

4.1.2.1 Metric

The clusters generation requires the definition of a metric for the evaluation of the distances among service elements. These distances can be calculated in advance and saved in a $|R| \times |R|$ matrix called D , where R is the set of all the service elements ($R = A_R \cup E_R \cup C_R$). The element (h, k) of this matrix represents the average distance between the service elements h and k . The computation of these distances is illustrated in figure 4.1, where the dashed lines are *shortest paths*. The average distance d_{AB} between two required vertices A and B , for example, is the average between the cost of the shortest path from A to B and the cost of the shortest path from B to A .

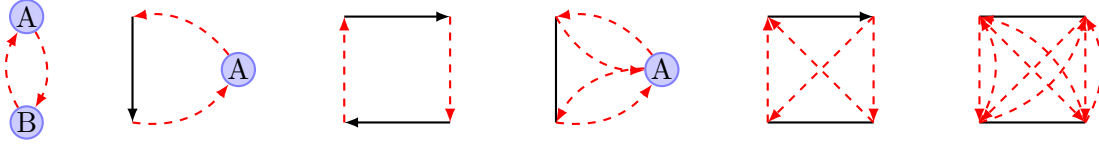


Figure 4.1: Computation of the average distance between service elements.

Let C_j be a cluster and h a service element, the distance between C_j and h is defined as the arithmetic mean of the average distances between h and each other element i in C_j :

$$d_{C_j h} = \frac{1}{|C_j|} \sum_{i=1}^{|C_j|} d_{ih} \quad (4.1)$$

4.1.2.2 Clusters generation

Algorithm 10: Clustering

Require: The graph G and the distance metric matrix D

Ensure: The set of clusters X

- 1: {Seeds selection}
- 2: Choose a random seed s , insert it in the first cluster C_1 and label s as chosen.

```

3: for  $i = 2$  to  $|K|$  do
4:    $s = \arg \max_{k \in R | k \text{ not labeled}} \{d_{C_j k} | j < i\}$ .
5:   Insert  $s$  in the cluster  $C_i$  and label  $s$  as choosen.
6: end for
7: {Fill the clusters}
8: Set the capacity of the cluster  $j$  to  $\frac{j}{|K|}Q$ .
9: for  $i = 1$  to  $|K|$  do
10:  {Insert in the cluster  $C_i$  the elements closest to the ones already present in it}
11:  while The capacity of the cluster  $C_i$  is not saturated do
12:    $s = \arg \min_{k \in R | k \text{ not labeled}} \{d_{C_i k}\}$ .
13:   Insert  $s$  in the cluster  $C_i$  and label  $s$  as choosen.
14:  end while
15: end for
16: {Insert remaining elements}
17: Restore the original capacities.
18: Sort the not labeled elements of  $R$  by non increasing demand.
19: for all Not labeled element  $s \in R$  do
20:   $c = \arg \min_{j \in X | j \text{ is compatible with } s} \{d_{C_j s}\}$ 
21:  if  $c$  exists then
22:   Insert  $s$  in the cluster  $C_c$  and label  $s$  as choosen.
23:  else
24:   Create a new cluster  $C'$ , insert  $s$  in  $C'$  and label  $s$  as choosen.
25:  end if
26: end for

```

The step 8 is necessary to avoid a penalization of the last clusters. Without this step, in fact, the last clusters could be filled with elements very far from each other. In the step 24 a new cluster is generated, in this case the algorithm returns a set of clusters X with $|X| > |K|$.

4. AN UPPER BOUND

4.2 Local Search

The local search algorithm is an iterative procedure moving from a solution to a better one in the space of candidate solutions until a given condition is reached (no more improving solutions or time limit). The search starts from a candidate solution (the initial solution described in section 4.1) and iteratively moves to a neighbor solution. A neighbor solution is obtained through a move named in the sequel *switch*($n, 1$). This move involves two clusters (C_i and C_j) and swaps n required elements of C_i with one required element of C_j . The move works on the clusters and not directly on the routes. After the swap operation, the new routes (and consequently the new cost of the solution) are obtained by solving two *GRP* problems on C_i and on C_j . Both the clusters and the required elements to swap are randomly chosen. The procedure ends when no improvements for γ consecutive swap operations occur. In the computational experiments we randomly choose between *swap*(2, 1) and *swap*(1, 1) at each iteration. The procedure is illustrated in the algorithm 11.

Algorithm 11: LocalSearch

Require: A set of clusters X

Ensure: A solution S

```
1:  $I = 0$ 
2: while  $I < \gamma$  do
3:    $C_i =$ random cluster in  $X$ ;
4:    $C_j =$ random cluster in  $X$  different from  $C_i$ ;
5:    $n = \text{random}\{1, 2\}$ ;
6:    $X' = \text{swap}(n, 1, C_i, C_j)$ ;
7:   calculate the total cost of  $X'$ ;
8:   if  $\text{cost}(X') < \text{cost}(X)$  then
9:      $I = 0$ ;
10:     $X = X'$ ;
11:   else
12:      $I = I + 1$ ;
13:   end if
14: end while
```

In the step 7, the total cost of the new clustering X' is calculated by optimally solving a *GRP* problem on C_i and C_j in order to obtain the new optimal routes and summing the costs of these routes to the ones of the unchanged clusters. The algorithm $swap(n, 1, C_i, C_j)$ is detailed in 12.

Algorithm 12: Swap

Require: n, C_i, C_j

Ensure: A new clustering X'

- 1: $L = n$ random elements of C_i ;
 - 2: $e = 1$ random element of C_j ;
 - 3: **if** $load(C_i) - load(L) + load(\{e\}) \leq Q$ AND $load(C_i) - load(\{e\}) + load(L) \leq Q$
then
 - 4: $X' = X \setminus \{C_i, C_j\}$;
 - 5: $C'_i = C_i \setminus L \cup \{e\}$;
 - 6: $C'_j = C_j \setminus \{e\} \cup L$;
 - 7: $X' = X' \cup C'_i \cup C'_j$;
 - 8: **else**
 - 9: $X' = X$;
 - 10: **end if**
-

With the notation $load(L)$ we indicate the total demand in L (where L is a set of required elements).

4. AN UPPER BOUND

5

The B&C Algorithm

Before the step by step branch and cut algorithm description, a strategy managing a particular set of inequalities (*cut pool*) and a simple procedure generating capacity and odd edge cutset constraints at the root node of the search tree are illustrated below.

5.1 Cut pool management

An iteration of the branch and cut algorithm involves the selection of a subproblem from the list of active subproblems and the addition of violated constraints and valid inequalities to this subproblem. The set containing these violated constraints and valid inequalities is called *cut pool*. It is cleaned every 50 iterations by eliminating inequalities with slack variables more than ϵ or dual variables less than ϵ , where $\epsilon = 10^{-6}$ is a tolerance.

5.2 Root node generation

The root node problem is composed by the objective function 2.1, the constraints from 2.2 to 2.6, the connectivity constraints 2.7 for the R-Sets (see 2.1) and the odd-edge and capacity constraints 3.3 and other connectivity constraints for some subsets of nodes generated by the routine 13.

Algorithm 13: Subsets generation

- 1: $W = \{1\}$
- 2: **while** $W \neq V$ **do**

5. THE B&C ALGORITHM

```
3:  $S = V \setminus W$ 
4: if  $\alpha(S) > 1$  then
5:   generate constraint 3.1 for  $S$ 
6: else if  $E_R(S)$  is odd then
7:   generate constraint 3.2 for  $S$ 
8: end if
9:   generate constraint 2.7 for  $S$ 
10: Add to  $W$  those nodes adjacent to at least one node of  $W$ 
11: end while
12: for all  $i \in C$  do
13:   if  $E_R(\{i\})$  is odd then
14:     generate constraint 3.2 for  $\{i\}$ 
15:   end if
16: end for
```

5.3 The Branch and Cut algorithm

A simple outline of the overall algorithm is provided in the algorithm 14.

Algorithm 14: Branch & Cut

```
1: Compute an upper bound  $\bar{\lambda}$  on the optimal solution value  $\lambda^*$  by using the local-  
   search procedure described in chapter 4.  
2: Define a relaxed MCGRP formulation as described in section 5.2 and eliminating  
   integrality constraints. Insert the resulting subproblem in a list  $L$ .  
3:  $I = 0$   
4: if  $L$  is empty then  
5:   STOP.  
6: else  
7:   Extract a subproblem from  $L$ .  
8: end if  
9: Solve the subproblem. Let  $\lambda$  be the solution value. If  $\lambda \geq \bar{\lambda}$ , go back to step 4.  
10: Identify constraints 2.7 through the heuristic algorithm 1.
```

5.3 The Branch and Cut algorithm

11: **if** No constraints 2.7 are identified by the heuristic algorithm 1 **then**
12: Identify constraints (2.7) through the exact algorithm 2.
13: **end if**
14: Identify constraints 3.1 through the algorithm 3.
15: **if** No constraints 2.7 are identified by the heuristic algorithm 1 **then**
16: Identify constraints (2.7) through the exact algorithm 2.
17: **end if**
18: **if** No inequalities of type 2.7 and 3.1 has been identified **then**
19: go to step 29.
20: **else**
21: Add the violated inequalities identified to the cut pool and to the current sub-
 problem.
22: **end if**
23: $I = I + 1$
24: **if** $I > 50$ **then**
25: Remove from the cut pool and from the current subproblem the constraints with
 slack variables more than ϵ or dual variables less than ϵ , where $\epsilon = 10^{-6}$ is a
 tolerance
26: $I = 0$
27: **end if**
28: Go back to step 9
29: Generate two son subproblems by branching on a fractional variable. Select the
 branching variable by considering the following order: $x_{ij}^k \in \{0, 1\}$, $z_i^k \in \{0, 1\}$,
 $y_{ij}^k \in \mathcal{Z}_+$. Insert the subproblems in L and go back to step 4.

5. THE B&C ALGORITHM

6

Computational experiments

Computational experiments have been carried out on a dataset of 34 instances with an Intel E2140 processor clocked at 1.60 GHz with 1.75 Gbyte RAM. The branch-and-cut algorithm has been coded in *Java 1.6*, by using ILOG CPLEX library, release 12.2. A time limit of four hours has been imposed to the computations by CPLEX, so that valid lower and upper bounds for the MCGRP are detected whenever the algorithm stops without satisfying the termination criterion (CPLEX optimality gap equal to zero). Note that the mixed integer problem cuts of CPLEX are active.

6.1 A First Illustration

In order to illustrate the complexity of the problem, we consider the MCGRP instance defined by the graph G depicted in Figure (6.1) and a fleet of $m = 4$ vehicles with capacity $Q = 10$. The optimal solution was obtained by running the complete MCGRP formulation (2.11). All the non-empty subsets of $C = \{1, 2, 3, 4, 5, 6\}$ were automatically generated and all the related connectivity constraints were introduced into the MCGRP model. CPLEX was able to find the optimal solution within a time of 0.42 seconds by exploring a search tree of 235 nodes, while our branch and cut algorithm reached the optimal solution in 0.09 seconds at the root node and by globally adding 11 inequalities. The optimal MCGRP solution consisting of 4 routes is shown in Figure (6.2), where solid lines indicate the links serviced in the route, dashed lines indicate the deadheading links and dotted lines represent the vertices that are not serviced. We remark that for instances a bit larger than the previous, with regard to the number of vertices

6. COMPUTATIONAL EXPERIMENTS

and links, CPLEX provides neither a lower bound nor an upper bound within a large computational time (i.e., more than 4 hours). Our branch and cut algorithm represents a first effective effort to approach the problem through an exact method, as shown in the following.

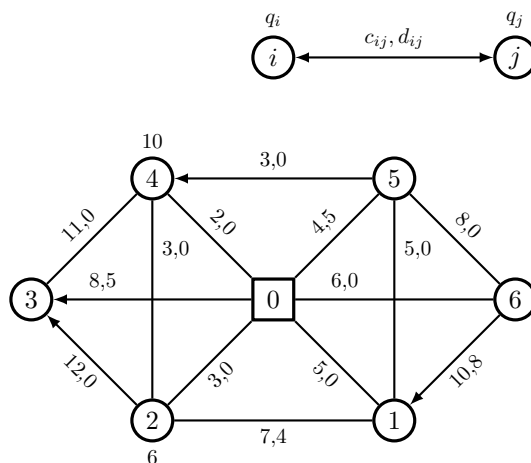


Figure 6.1: $G = (V, E, A)$.

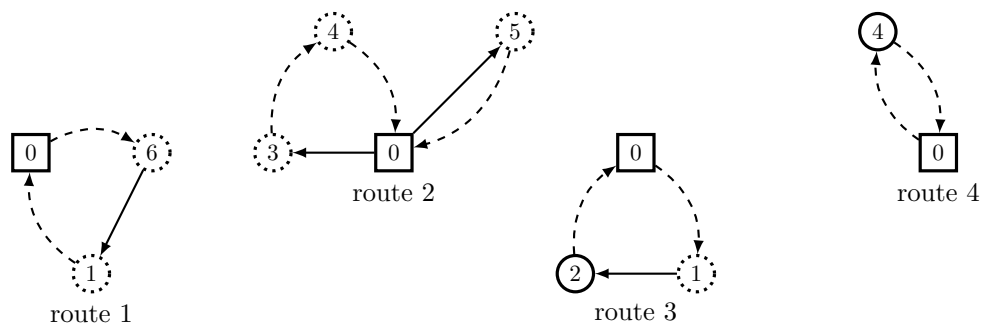


Figure 6.2: Optimal solution.

6.2 The Instances

We derived a dataset of new instances for the MCGRP from 34 instances called *mval* and designed by Belenguer et al. (2006) for the mixed capacitated ARP. The original instances contain 24-50 nodes and 43-138 links, all required. Each of them was converted into an *mgval* one (*g* like *general*), with the same total demand, as follows. Let $\pi = |A_R \cup E_R|$ be the number of required links for an *mval* instance. The demands on

$\lceil 0.1\pi \rceil$ randomly selected links were transferred to $\lceil 0.1\pi \rceil$ randomly selected nodes. So, for example, if an *mval* instance has 24 nodes and 55 required links, the corresponding *mgval* instance has 6 required nodes, 18 non-required nodes, 49 required links, and 6 non-required links.

6.3 Numerical Results

The numerical results obtained on the above sets of instances are reported in table 6.1. The column headings are defined as follows:

<i>file</i>	: instance name;
$ V $: number of vertices of the graph;
$ A \cup E $: number of arcs and edges of the graph;
<i>m</i>	: number of vehicles;
LB	: lower bound at the root of the search tree;
UB	: initial upper bound;
CON	: number of added connectivity constraints;
SUR	: number of added surrogate capacity inequalities;
λ	: solution value (final upper bound);
GAP%	: percentage gap computed with respect to the optimal value;
$\frac{ LB - \lambda }{\lambda}$: integer ratio;
Nodes	: number of nodes in the search tree.

Observe that the GAP% is defined as CPLEX parameter in the following way: $|BestNode - BestInteger| / (10^{-10} + |BestInteger|)$. *BestNode* is the best objective value returned by CPLEX within the time limit and *BestInteger* is the best integer solution value. Whenever the model is solved to optimality, GAP% is equal to zero and we report in column λ the optimal value (an asterisk denotes this case). Otherwise, *BestNode* is computed as the minimum value of the objective function of all remaining unexplored nodes.

6. COMPUTATIONAL EXPERIMENTS

The number of instances solved to optimality is equal to 12. Specifically, we obtained optimal solutions for instances with a number of vehicles less than or equal to 4 for small and large graphs. In general, some branching is required to reach such optimal solutions, with the exception of two instances that were solved at the root node. For the remaining instances lower and upper bounds are reported in table 6.1.

Observe that connectivity constraints were added in each instances in order to ensure the feasibility of the solution. Surrogate capacity inequalities considerably contribute to obtain a feasible or optimal solution. For instances, in the instance *mgval2b* the number of identified surrogate inequalities is huge. Only in 6 instances they are irrelevant.

In figures 6.3 and 6.4 we can observe that for the instances in which we have the optimal solution, the optimal cost is very close to the lower bound (except for *mgval9c*). For the other instances the final feasible solution has a cost close to the upper bound. This suggest that is not convenient work on the lower bound improvement, but reducing the upper bound can help to reduce the computational effort by cutting the branch and cut tree.

<i>file</i>	$ V $	$ A \cup E $	m	LB	UB	CON	SUR	λ	GAP%	$\frac{LB-\lambda}{\lambda}$	Nodes
<i>mgval1a</i>	24	55	2	200.00	208	1475	0	202*	0.00	1.00	4
<i>mgval1b</i>	24	51	3	236.00	287	254	17	236*	0.00	0.00	5
<i>mgval1c</i>	24	53	8	259.70	—	15322	28	309	15.97	19.00	8373
<i>mgval2a</i>	24	44	2	305.00	379	1701	23	305*	0.00	0.00	1
<i>mgval2b</i>	24	52	3	374.70	393	10231	459	381*	0.00	1.69	2972
<i>mgval2c</i>	24	49	8	332.00	630	20895	6600	617	45.83	85.84	10195
<i>mgval3a</i>	24	48	2	104.00	115	1444	10	105*	0.00	0.96	4
<i>mgval3b</i>	24	45	3	137.00	155	327	6	137*	0.00	0.00	1
<i>mgval3c</i>	24	43	7	142.00	182	10891	15851	165	12.49	16.20	16466
<i>mgval4a</i>	41	95	3	543.00	590	109918	7493	560	1.84	3.13	6836
<i>mgval4b</i>	41	102	4	585.00	696	112606	2773	590	0.85	0.85	2056
<i>mgval4c</i>	41	103	5	575.00	744	79337	2579	641	10.3	11.48	1479
<i>mgval4d</i>	41	104	9	601.00	872	20063	254	872	31.08	45.09	1
<i>mgval5a</i>	34	96	3	547.00	599	7853	86	547*	0	0.00	7
<i>mgval5b</i>	34	91	4	523.00	616	66363	4523	542	3.51	3.63	5340
<i>mgval5c</i>	34	98	5	646.00	750	47809	468	663	2.56	2.63	4715
<i>mgval5d</i>	34	92	9	616.00	866	34260	640	866	28.87	40.58	71
<i>mgval6a</i>	31	69	3	287.00	321	776	17	287*	0	0.00	3
<i>mgval6b</i>	31	66	4	296.00	323	1987	1	296*	0	0.00	1
<i>mgval6c</i>	31	68	10	304.00	440	22600	1055	440	30.91	44.74	681
<i>mgval7a</i>	40	86	3	336.50	383	21744	2	341*	0	1.34	27
<i>mgval7b</i>	40	91	4	388.00	431	2151	0	388*	0	0.00	1
<i>mgval7c</i>	40	90	9	369.00	473	15101	188	473	21.99	28.18	17
<i>mgval8a</i>	30	96	3	566.00	603	7474	17	566*	0	0.00	1
<i>mgval8b</i>	30	91	4	482.00	552	48810	532	484*	0	0.41	385
<i>mgval8c</i>	30	83	9	511.00	720	27424	475	720	29.03	40.90	30
<i>mgval9a</i>	50	132	3	404.60	455	47696	284	407*	0	0.60	52
<i>mgval9b</i>	50	120	4	393.00	473	94639	2150	396	0.76	0.76	1719
<i>mgval9c</i>	50	125	5	396.00	455	69236	196	455*	12.97	14.90	1
<i>mgval9d</i>	50	131	10	436.00	573	9161	127	573	23.91	31.42	1
<i>mgval10a</i>	50	138	3	593.00	632	15214	20	593*	0	0.00	1
<i>mgval10b</i>	50	134	4	570.00	666	92362	65	572	0.35	0.35	971
<i>mgval10c</i>	50	136	5	563.50	640	85547	321	640	11.95	13.58	1
<i>mgval10d</i>	50	129	10	536.00	744	11586	257	744	27.96	38.81	1

Table 6.1: Computational results for *mgval* instances

6. COMPUTATIONAL EXPERIMENTS

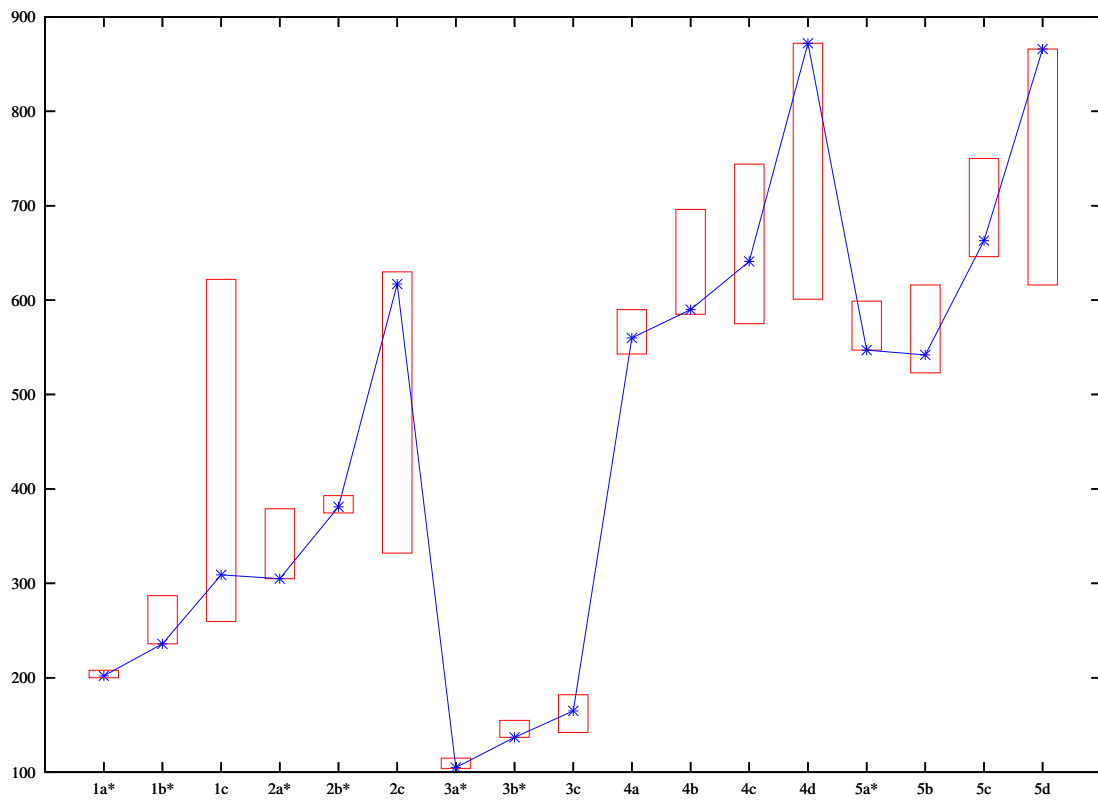


Figure 6.3: Comparison among Lower Bound, Upper Bound and final cost - Instances mgval1A to mgval5D

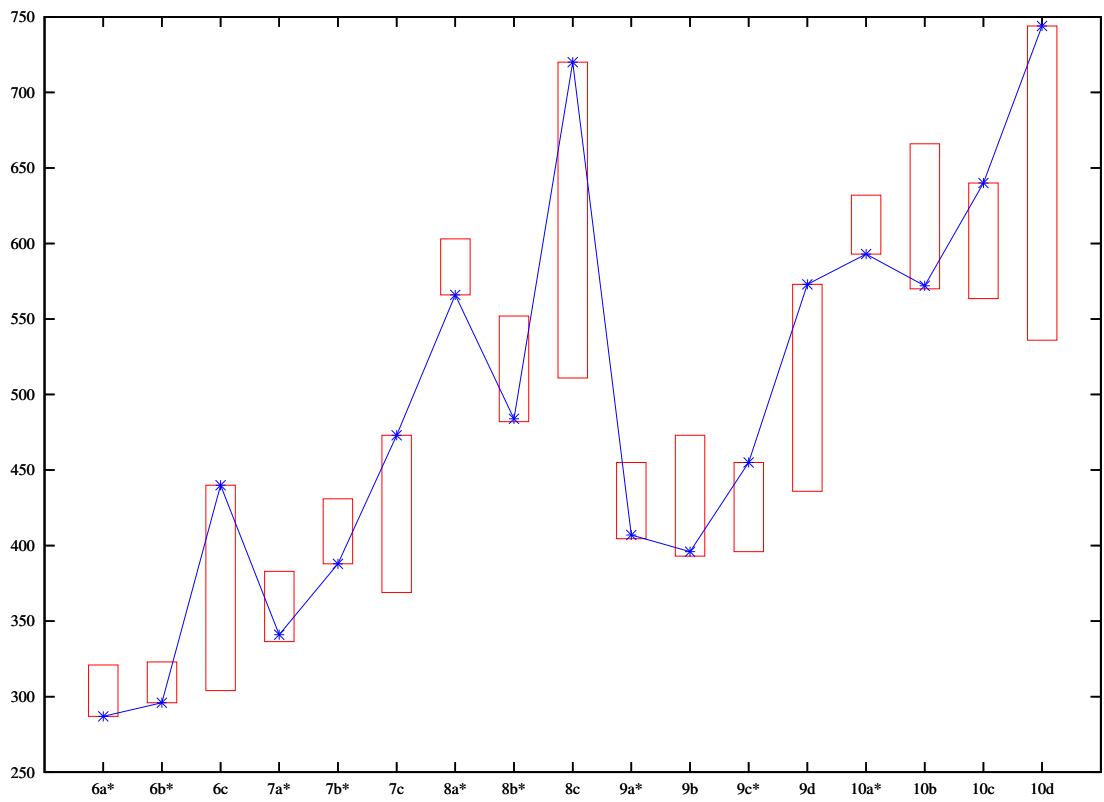


Figure 6.4: Comparison among Lower Bound, Upper Bound and final cost - Instances mgval6A to mgval10D

6. COMPUTATONAL EXPERIMENTS

7

Conclusions

This thesis deals with the problem of optimally solve the Mixed Capacitated General Routing through a branch and cut algorithm. In Chapter 1 we give a brief introduction on the problem, highlighting its practical impact in many real life circumstances. Moreover we expose the literature on this problem, observing that there are no other works regarding any exact approach to solve it.

In Chapter 2 we introduce an integer programming model for the MCGRP, based on three-index link variables and two-index node variables. This model contains an huge number of constraints due to the presence of the classical connectivity constraints derived by the TSP problem. For this reason, in Chapter 3 the relevant separation procedure are described in order to identify violations of such inequalities. In the same chapter is described how to extend some well-known valid inequalities for the Undirected Capacitated Arc Routing polyhedron to the Mixed Capacitated General Routing polyhedron with the aim of improving the lower bound at the root node of the search tree and cutting off the fractional solutions. This issue is addressed by designing heuristic and exact separation procedures.

In Chapter 4 an upper bound procedure is described to obtain either a good feasible solution of the problem helping the pruning process in the search tree or a good near-optimal solution for big instances whether the branch and cut algorithm is not able to achieve an optimal solution.

The Chapter 5 describes the overall branch and cut algorithm and Chapter 6 presents the computational results on a set of instances derived from the mixed capacitated ARP and adapted to the MCGRP. The results show that our algorithm is able

7. CONCLUSIONS

to find the optimal solution on about half of the instances and a good feasible solution on the remaining instances. We observe that the complexity of the problem increases with the number of vehicles needed to service all the required elements, whereas the optimal solutions are very close to the lower bounds. These considerations suggest to concentrate the future efforts towards improving the upper bound procedures.

References

- J. Albiach, J.M. Sanchis, and D. Soler. An asymmetric tsp with time windows and with time-dependent travel times and costs: An exact solution through a graph transformation. *European Journal of Operational Research*, 189(3):789–802, 2008. 3
- P. Augerat, J.M. Belenguer, E. Benavent, A. Corberán, and D. Naddef. Separating capacity constraints in the cvrp using tabu search. *European Journal of Operational Research*, 106:546–557, 1998. 19
- R. Baldacci and V. Maniezzo. Exact methods based on node-routing formulations for undirected arc-routing problems. *Networks*, 47(1):52–60, 2006. viii, 2
- J.M. Belenguer and E. Benavent. The capacitated arc routing problem: Valid inequalities and facets. *Computational Optimization and Applications*, 10(2):165–187, 1998. 17
- J.M. Belenguer and E. Benavent. A cutting plane algorithm for the capacitated arc routing problem. *Computers and Operations Research*, 30(5):705–728, 2003. 17, 19
- J.M. Belenguer, E. Benavent, P. Lacomme, and C. Prins. Lower and upper bounds for the mixed capacitated arc routing problem. *Computers and Operations Research*, 33(12):3363–3383, 2006. 44
- E. Benavent, A. Corberán, and J.M. Sanchis. *Arc routing: Theory, solutions and applications*, chapter Linear programming based methods for solving arc routing problems., pages 231–275. Kluwer Academic Publishers, Boston, USA, 2000. 16
- E.G. Coffman, Garey M.R., and Johnson D.S. *Approximation Algorithms for NP-Hard Problems*, chapter Approximation algorithms for bin packing: A survey., pages 46–93. PWS Publishing, Boston, USA, 1996. 5
- A. Corberán and J.M. Sanchis. The general routing problem polyhedron: Facets from the rpp and gtsp polyhedra. *European Journal of Operational Research*, 108(3):538–550, 1998. viii, 3
- A. Corberán, A.N. Letchford, and J.M. Sanchis. A cutting plane algorithm for the general routing problem. *Mathematical Programming, Ser. A*, 90(2):291–316, 2001. viii, 3, 22
- A. Corberán, A. Romero, and J.M. Sanchis. The mixed general routing polyhedron. *Mathematical Programming, Ser. A*, 96(1):103–137, 2003. viii, 3
- A. Corberán, G. Mejía, and J.M. Sanchis. New results on the mixed general routing problem. *Operations Research*, 53(2):363–376, 2005. ix, 3
- A. Corberán, I. Plana, and J.M. Sanchis. A branch&cut algorithm for the windy general routing problem and special cases. *Networks*, 49(4):245–257, 2007. 3

REFERENCES

- A. Corberán, I. Plana, and J.M. Sanchis. The windy general routing polyhedron: A global view of many known arc routing polyhedra. *SIAM Journal on Discrete Mathematics*, 22(2):606–628, 2008. 3
- G. Cornuejols and F. Harche. Polyhedral study of the capacitated vehicle routing problem. *Mathematical Programming*, 60:21–52, 1993. 2
- M. Fischetti, J.J. Salazar, and P. Toth. A branch-and-cut algorithm for the symmetric generalized traveling salesman problem. *Operations Research*, 45(3):378–394, 1997. 13
- B.L. Golden and R.T. Wong. Capacitated arc routing problems. *Networks*, 11(3):305–315, 1981. 2
- J.C.A. Gutiérrez, D. Soler, and A. Hervás. The capacitated general routing problem on mixed graphs. *Revista Investigacion Operacional*, 23(1):15–26, 2002. ix, 4, 27
- J.K. Lenstra and A.H.G. Rinnooy Kan. On general routing problems. *Networks*, 6(3):273–280, 1976. viii, ix, 2
- A.N. Letchford. New inequalities for the general routing problem. *European Journal of Operational Research*, 96(2):317–322, 1996. viii, 3
- A.N. Letchford. The general routing polyhedron: A unifying framework. *European Journal of Operational Research*, 112(1):122–133, 1999. viii, 3
- H. Longo, M. Poggi de Aragão, and E. Uchoa. Solving capacitated arc routing problems using a transformation to the cvrp. *Computers and Operations Research*, 33(6):1823–1837, 2006. viii, 2
- L. Muyldermans, P. Beullens, D. Cattrysse, and D. Van Oudheusden. Exploring variants of 2-opt and 3-opt for the general routing problem. *Operations Research*, 53(6):982–995, 2005. viii, 3
- C.S. Orloff. A fundamental problem in vehicle routing. *Networks*, 4(1):35–64, 1974. 2
- M.W. Padberg and M.R. Rao. Odd minimum cut-sets and b -matchings. *Mathematics of Operations Research*, 7(1):67–80, 1982. 22
- R. Pandit and B. Muralidharan. A capacitated general routing problem on mixed networks. *Computers and Operations Research*, 22(5):465–478, 1995. ix, 3
- W.L. Pearn, A. Assad, and B.L. Golden. Transforming arc routing into node routing problems. *Computers and Operations Research*, 14(4):285–288, 1987. viii, 2
- C. Prins and S. Bouchenoua. *Studies in Fuzziness and Soft Computing, Recent Advances in Memetic Algorithms*, volume 166, chapter A Memetic Algorithm Solving the VRP, the CARP and general routing problems with nodes, edges and arcs., pages 65–85. Springer, 2005. 4
- G. Reinelt and D.O. Theis. On the general routing polytope. *Discrete Applied Mathematics*, 156(3):368–384, 2008. 3
- D. Soler, E. Martínez, and J.C. Micó. A transformation for the mixed general routing problem with turn penalties. *Journal of the Operational Research Society*, 59(4):540–547, 2008. 3