Università della Calabria

Facoltà di Ingegneria

Dipartimento di Elettronica, Informatica e Sistemistica (DEIS)

DOTTORATO DI RICERCA IN RICERCA OPERATIVA, XXII CICLO (2007-09)

Mixed General Routing Investigations

Coordinatore: Prof. Lucio Grandinetti

Dottorando: Salvatore Capolupo

Relatore: Prof. Roberto Musmanno

Correlatore: Ing. Demetrio Laganà

Settore Scientifico-Disciplinare (SSD) : **MAT/09**

UNIVERSITÀ DELLA CALABRIA

FACOLTÀ DI INGEGNERIA

Dipartimento di Elettronica, Informatica e Sistemistica

Dottorato di Ricerca in Ricerca Operativa
XXII Ciclo (2007-09)
Tesi di dottorato

Mixed Capacitated General Routing Problem Investigations

Coordinatore: Prof. Lucio Grandinetti
Relatore: Prof. Roberto Musmanno
Correlatore: Prof. Demetrio Laganà

Dottorando: Salvatore Capolupo

Settore Scientifico disciplinare: **MAT/09**

CONTENTS

LIST OF FIGURES

LIST OF TABLES

1. INTRODUCTION

Routing problems typically arise in several areas of distribution management and logistics, and their practical significance is widely known. The common objective of such problems is addressed to satisfy the total demand localized over a logistical network, by constructing a set of minimum feasible routes (i.e. with minimum traveling time) starting from the depot and ending into it, and servicing a subset of required links or nodes in the network. In the node-routing problems the demand (or service) occurs in the nodes, while in the arc-routing problems is assumed to be along the arcs (or edges).

In the general routing problems (GRPs) both two features are merged in a single problem. GRP can be exploited to model real-life problems, like optimal routing for garbage collection over a road network: this is a very practical impact problem, in which companies are interested to optimize total travel time in vehicles employed for the collections of garbage bins. Many practical logistic problems may be studied by resorting to the arc and node-routing linear programming models. This thesis has been outlined in the followings sections: in the first section some essential scientific literature (9) has been presented; in the second section a mathematical formulation of the Mixed Capacitated General Routing Problem (MC-GRP) has been described and critically analyzed. In the third section a branch and cut algorithm has been proposed and some generalized polyhedral results have been discussed and presented. Finally the computational results and complexity of the proposed algorithm have been illustrated.

1.1 Literature Review

The MCGRP (also know as CGRP-m in [7]) is a routing problem that aims to minimize the total transportation cost of a set of routes servicing all required link and nodes. Each route starts from depot and ends into it by collecting a subset of required links and nodes without exceeding its capacity. We consider an homogeneous fleet of vehicles, with same capacity for each of them. In the scientific literature not many papers are related to the MCGRP: moreover in most of the cases, authors take into account capacitated or mixed graph features separately. Otherwise the MCGRP includes many well-known routing problems only as special cases. Here we propose a fast overview of the main results produced over this kind of problem until now. Orloff in [3] proposed the first algorithm for GRP on symmetric graph: it provides an unified approach to node-routing and arc-routing problems, useful for making tractable effective big-sized problem of this kind. The classical Traveling Salesman Problem (TSP) and the Chinese Postman Problem (CPP) are shown to be special limiting cases of the General Routing Problem: this implies that GRP is also a NP-Hard problem. Another important first result for GRP refers to separation problems associated with connectivity and R-odd cut inequalities: these are solvable in polynomial time, by means of max-flow calculations and the Padberg & Rao procedure (see [11], [1]). This result can be easily extended to the MGRP ([9]): in the course of the algorithm additional inequalities of the above mentioned classes are generated as they are checked as violated. When this is no longer possible, and the LP solution is still not integral, we invoke branch and bound. If the resulting integer solution is feasible for the MGRP, it is optimal. Otherwise, the procedure terminates with a tight lower bound, but no feasible MGRP solution. A heuristic procedure for the MCGRP was subsequently proposed in [4], with a single vehicle and working-hours constraints: this algorithm is based on route first-cluster second and its dual approach cluster first-partition second. Then Letchford in [16] showed how to transform the General Routing Problem (GRP) into a variant of the Graphical Travelling Salesman Problem (GTSP), and found also some important valid inequalities for the GRP polyhedron. In [1] author remarks other valid inequalities for the GRP, and he also explains how in Mixed Chinese Postman Problem (MCCP) we can define the set of feasible solutions by some specific conditions. Besides, it is shown that we can use without distinction two or one integer variable(s) for representing edge crossing. Between the most important contributions of last years, many work was done by Corberan, Sanchis et al.: in [6] they described a new family of facet-inducing inequalities for the GRP, which seem to be very useful for solving GRP and RPP instances. Further, they shown new classes of facets obtained by composition of facet-inducing inequalities. In [7] it was proposed an improved heuristic procedure than [4], proved by some computational results: in particular they solved successfully until 50 nodes and 98 link instances of mixed-graph, also capacitated. However this approach does not take in account transforming mixed graph instance into an equivalent ACVRP one, and use any exact procedure on this for solving original problem. Meanwhile [9] and [6] point attention about GRP polyhedron, finding important theoretical results. In particular, they proposed a cutting-plane algorithm with new separation procedures for three class of inequalities: extensive computational experiments over various sets of instances was included. Similarly in [5] authors proposed for GRP a very efficient local-search, in which their computational experiments produced high-quality solutions within limited computation time. Some authors had computed some good bounds for this problem: i.e., in [8] a lower bound is computed with a cutting-plane procedure, also invoking a branch-and-bound procedure. Instead upper-bound is computed exploiting a heuristic or meta-heuristic procedure.

1.2 Contributions.

In this section, we summarize the main contributions of this thesis.

We propose a MIP formulation for the problem using three-index variables: it has advantages of a good mathematical tractability, but for ”big” instances it could be very time-consuming and not usable in practice. So this was only a start point for our work, that aimed us to relax some complicating constraints (including integer and so called connectivity inequalities).

We implemented a GRASP-based heuristic (Greedy Randomized Adaptive Search Procedure) to obtain an upper-bound for the MCGRP. Our approach uses a cluster first-route second for making first routes, which are trivially feasible by construction. A distance definition between cluster and required element helps us to execute a post-optimization procedure, recombing routes and avoiding having some of them exceeding capacity. The variation of the number of vehicle m^* offers the flexibility of constructing feasible solution into the variable neighborhood. Finally we propose a branch&cut algorithm to optimality solve several random-generated instances of the MCGRP: this was performed by extending to the MCGRP classical connection, co-circuit and balanced-set inequalities. An in-deep analysis of our algorithm's performances is faced by studying the improving gap obtained for each class of violated constraints.

Part I

PROBLEM DESCRIPTION.

2. MATHEMATICAL FORMULATIONS FOR THE MCGRP

2.1 Definitions.

Let be:

- $G = (V, E, A)$ a mixed graph defined over a set of vertices V, a set of edges E and a set of arcs A;
- $C = V \setminus \{v_{depot}\}$ the customer set, where v_{depot} represents the node depot;
- $C_R \subseteq C$ the required-customer set of nodes, with non-negative demands $q_i > 0$;
- $A_R \subseteq A$ the required-customer set of arcs, with non-negative demands $d_{ij} > 0$;
- $E_R \subseteq E$ the required-customer set of edges, with non-negative demands $d_{ij} > 0$;
- $R = C_R \cup E_R \cup A_R$ the set of required nodes, arcs and edges. In the following we will refer to each element of R as ”required element”.
- $K = \{1, \dots, m^*\}$ the set of vehicle indexes, with some capacity Q.

Definition 1: We define: $m = \left\lceil \frac{\sum_{(i,j)\in E_R \cup A_R} d_{ij} + \sum_{i \in C_R} q_i}{Q} \right\rceil$ a lower-bound for m^* ($m \leq m^*$).

Observation 1: Finding the minimum number m^* of vehicles to service all the required elements can be reached by optimality solving the following 1-Bin packing problem:

$$\min m^* = \sum_{k \in M} y^k \tag{2.1}$$

$$\sum_{i \in C_R} z_i^k + \sum_{(i,j) \in E_R \cup A_R} x_{ij}^k \leq Q \cdot y^k, \forall k \in M \tag{2.2}$$

$$\sum_{k \in M} x_{ij}^k = 1, \, \forall (i,j) \in E_R \cup A_R \tag{2.3}$$

$$\sum_{k \in M} z_i^k = 1, \, \forall i \in C_R \tag{2.4}$$

$$x_{ij}^k, z_i^k \in \{0,1\} \tag{2.5}$$

1-Bin Packing is a well-know NP-Hard class problem, which can be solved exactly only for small instances, or alternatively exploiting (meta)heuristics. Note that $|M|$ represents the maximum vehicle number, and considering that this value can't be greater than cardinality of all required elements, we can assign $|M| = |C_R| + |E_R| + |A_R|$.

Definition 2: Given a mixed-graph $G = (V, E, A)$ and an integer permutation $\sigma : I_v \rightarrow \mathbb{N}$ such that $\sigma(i) = j$ with $i \in I_v$ and $j \in \mathbb{N}$, and where I_v is the set of indices mapping all the vertices in V, a route is defined as:

$$\begin{aligned} \rho = \{(v_{\sigma(1)}, v_{\sigma(2)}), \ldots, (v_{\sigma(h-1)}, v_{\sigma(h)})) : \\ v_{\sigma(1)} = v_{\sigma(h)} \equiv v_{depot} \wedge \\ (v_{\sigma(i)}, v_{\sigma(j)}) \in E \cup A \; \forall i, j \in I_v \wedge \\ v_{\sigma(i+1)} = v_{\sigma(i)} \; \forall i \in I_v \setminus \{1, h+1\}\} \end{aligned}$$

In fig 2.1 we show an example, where for sake of simplicity we used $\sigma(i) = i, \forall i \in I_v$

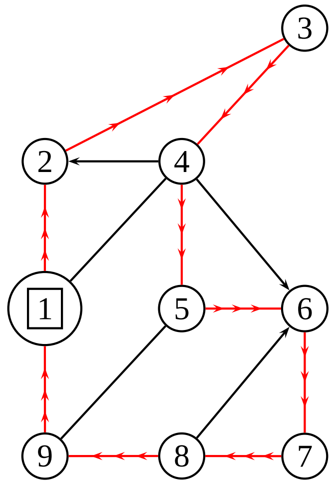


Fig. 2.1: Route example $\rho = \{(1,2)(2,3)(3,4)(4,5)(5,6)(6,7)(7,1)\}$;

2.1.1 Problem and objective.

The MCGRP generalizes many vehicle routing problems that have been studied in the last forty years, for which hundreds of papers have been written, either to give exact or heuristic procedures for their resolution and bounds.

These are specific characterizations of our problem, and we can cite as examples:

- if $A = \emptyset = E_R$ we have the Capacitated Vehicle Routing Problem(CVRP);
- if $A = \emptyset = C_R$ we have the Capacitated Arc Routing Problem(CARP);
- if $E = \emptyset = E_R$ we have the Asymmetric Capacitated Vehicle Routing Problem(ACVRP);
- if $k = 1$ we have the General Routing Problem(CVRP);

The Mixed Capacitated General Routing Problem can be formally defined as follows.

Definition 3: Let $G = (V, E, A)$ be a strongly connected mixed graph where:

- vertex $1 \in V$ represents the depot, and exists at least a customer c_i;
- each link $(i, j) \in E \in A$ has an associated non-zero cost c_{ij} (note that $c_{ii} = 0$ and $\forall (i, j) \notin E \in A$ $c_{ij} = \infty$);
- it exists a customer subset C_R such that each vertex $i \in C_R$ has got a positive demand $0 < q_i \leq Q$;
- it exists a customer subset E_R such that each edge $e = (i, j) \in E_R$ has got a positive demand $0 < q_e \leq Q$;
- it exists a customer subset A_R such that each vertex $a = (i, j) \in A_R$ has got a positive demand $0 < q_a \leq Q$;
- the sum of all demands $\sum_{i \in C_R} q_i + \sum_{(i,j) \in E_R \cup E_R} q_{ij}$ does not exceed Q, where Q is fixed and constant.

The objective is to find m tours $Q-$capacitated in G such that:

- each tour passes through node 1;
- all demands q_i, q_e, q_a are fully satisfied (i.e. no residual demands remains over a required component);
- each customer $i \in CR$, $a \in A$ and $e \in E$ are served by exactly one of the m tour;
- the sum of all demands $\sum_{i \in C_R} q_i + \sum_{(i,j) \in E_R \cup E_R} q_{ij}$ does not exceed Q;
- the sum of all costs is optimal (i.e. minimum of sum the costs over the links into activated routes).

2.1.2 Cutsets.

We define cutsets $\forall S \subset V$:

- $A^+(S) = \{(i,j) \in A, \forall i \in S,\ j \in V \setminus S\} = A(S : V \setminus S)$
- $A^-(S) = \{(j,i) \in A, \forall j \in V \setminus S,\ i \in S\} = A(V \setminus S : S)$
- $E^+(S) = \{(i,j) \in E, \forall i \in S,\ j \in V \setminus S\} = E(S : V \setminus S)$
- $E^-(S) = \{(j,i) \in E, \forall j \in V \setminus S,\ i \in S\} = E(V \setminus S : S)$
- $A_R^+(S) = \{(i,j) \in A_R, \forall i \in S,\ j \in V \setminus S\} = A_R(S : V \setminus S)$
- $A_R^-(S) = \{(j,i) \in A_R, \forall j \in V \setminus S,\ i \in S\} = A_R(V \setminus S : S)$
- $E_R^+(S) = \{(i,j) \in E_R, \forall i \in S,\ j \in V \setminus S\} = E_R(S : V \setminus S)$
- $E_R^-(S) = \{(j,i) \in E_R, \forall j \in V \setminus S,\ i \in S\} = E_R(V \setminus S : S)$
- $E(S) = E^+(S) \cup E^-(S)$
- $A(S) = A^+(S) \cup A^-(S)$
- $E_R(S) = E_R^+(S) \cup E_R^-(S)$
- $A_R(S) = A_R^+(S) \cup A_R^-(S)$
- $S_R = S \cap C_R$
- $\gamma_R(S) = E_R(S) \cup A_R(S) \cup S_R$

2.2 Variables.

We will use three-index variables, where superscript will always refer to k-route and subscript to (i,j) link (or i for a node).

2.2.1 Double-Edge variables.

This representation requires a very large number of variable: if we got a very large majority of edges (i.e. $|E| \gg |A|$) this could lead to very big models, whose could be computationally inefficient.

Service-link variable: x_{ij}^k

We define the binary variable $\forall k = 1, \ldots, m$:

$$x_{ij}^k = \begin{cases} 1 & \text{if } k\text{-vehicle serves link } (i,j) \in E \cup A \text{ ;} \\ 0 & \text{elsewhere.} \end{cases}$$

Service-link variable: y_{ij}^k

We define the binary variable $\forall k = 1, \ldots, m$:

$$y_{ij}^k = \begin{cases} 1 & \text{if } k\text{-vehicle crosses link } (i,j) \in E \cup A \text{ ;} \\ 0 & \text{elsewhere.} \end{cases}$$

Service-node variable: z_i^k

We define the binary variable $\forall k = 1, \ldots, m$:

$$z_{ij}^k = \begin{cases} 1 & \text{if } k\text{-vehicle serves node } i \in C_R \text{ ;} \\ 0 & \text{elsewhere.} \end{cases}$$

The number of total variables is here $2 \cdot |E| + |A| + |V|$, because we distinguish between straight (i.e. from i node to j) and reverse crossings (i.e. from j node to i) over every edges. In what following we will describe main conditions for our problem.

2.2.2 Parity and balanced-se conditions

Definition 4: Given a mixed graph $G = (V, E, A)$, we say a node $v \in V$ is even iff has got a even number of incident links (degree), otherwise node is odd. Similarly we define a node being R-even (resp. R-odd) iff has got a even (resp. odd) number of incident required links. If degree is equal to 0, then the node is conventionally even.

Definition 5: Given a mixed graph $G = (V,E,A)$, a node set $S \subseteq V$, an integer index $k \in K$ and an integer variable $\xi : L(S) \rightarrow \mathbb{N} \cup \{0\}$, with $L(S) = E(S) \cup A^+(S) \cup A^-(S)$, we say S is set-balanced iff satisfy the following:

$$\xi(A^+(S)) + \xi(A^-(S)) + \xi(E(S)) \leq u_S$$

$$u_S = |A^+(S)| + |A^-(S)| + E(S), \forall S \subset V$$

That is, if we consider contribution of every activated traveling-variable (first member of inequality) with respect to every possible link of the same set (second member), we have that first sum is greater or equal to u_S, then S is set-balanced (and vice-versa).

Here we report two simple examples for clarifying these two conditions.

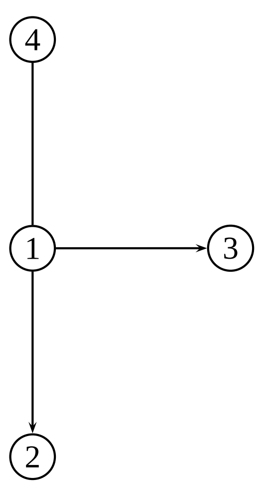


Fig. 2.2: Mixed-graph for parity and balanced-set examples

v	parity
1	R-odd, even
2	R-even, odd
3	R-odd, even
4	R-even, odd

Tab. 2.1: Parity for Fig. 2.2.2 nodes

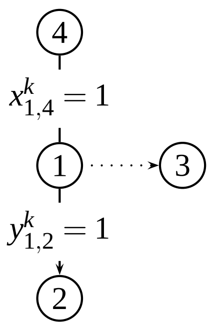


Fig. 2.3: Balanced-set example, where $x^k_{1,4} = 1$ and $y^k_{1,2} = 1$.

In represented graph in fig. 2.2.2 we've got situation represented in Table 2.1.

In mixed-graph represented in fig. 2.2.2 the balanced-set condition depends on activated variables: in fig. 2.3 is balanced, meanwhile in fig. 2.4 is unbalanced.

2.3 Constraints.

Here we will briefly describe the constraints for our problem. We need to minimize a cost function computed over all used routes, with the following requirements:

1. every service component must be served only once (*assignment*);
2. total quantity carried by every vehicle cant excess fixed capacity of that vehicle (*knapsack:*);

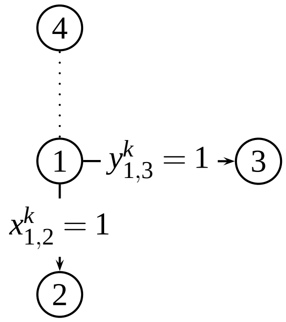


Fig. 2.4: Unbalanced-set example, where $x^k_{1,2} = 1$ and $y^k_{1,3} = 1$.

3. we must assure parity for every node of every route (*parity:*) ;
4. we must assure balancing for every node of every route (*balanced-set:*);
5. we must assure every route is connected (*connection:*);

We can express these constraints in mathematical form as follows.

2.3.1 Assignment

$$\sum_{k=1}^{m} (x^k_{ij} + x^k_{ji}) = 1, \forall (i, j) \in E_R \subseteq E \tag{2.6}$$

$$\sum_{k=1}^{m} x^k_{ij} = 1, \forall (i, j) \in A_R \subseteq E \tag{2.7}$$

$$\sum_{k=1}^{m} z^k_i = 1, \forall (i, j) \in C_R \subseteq V \tag{2.8}$$

Here we imposed three kind of constraints for each required edge (resp. arc and node), that is sum of these over all m routes must be equal to 1, so every required elements must be served only a time: the number of trips is supposed constant and equal to lower-bound given in 1.

2.3.2 Knapsack

$$\sum_{(i,j)\in E_R} d_{ij}(x_{ij}^k + x_{ji}^k) + \sum_{(i,j)\in A_R} d_{ij}x_{ij}^k + \sum_{i\in C_R} d_i z_i^k \leq Q, \forall k \in K \tag{2.9}$$

These constraints impose for each route that fixed capacity Q of every vehicle cant be exceeded for every route we consider.

2.3.3 Parity & balanced-set

We represent parity and balanced-set condition as a single group of constraints, where in first member we count the total number of activated arcs and in second member edges contribution. That assures that in

$$\sum_{\forall j:\ (i,j)\in A_R^+(i)} x_{ij}^k + \sum_{\forall j:\ (i,j)\in A^+(i)} y_{ij}^k - \sum_{\forall j:\ (j,i)\in A_R^-(i)} x_{ji}^k - \sum_{\forall j:\ (j,i)\in A^-(i)} y_{ji}^k =$$
$$\sum_{\forall j:\ (j,i)\in E_R^-(i)} x_{ji}^k + \sum_{\forall j:\ (j,i)\in E^-(i)} y_{ji}^k - \sum_{\forall j:\ (i,j)\in E_R^+(i)} x_{ij}^k - \sum_{\forall j:\ (i,j)\in E^+(i)} y_{ij}^k,$$
$$\forall i \in V, \forall k \in K$$

2.3.4 Connection

These constraints are used to assuring our tours are connected, that is every tour starts from depot and returns into it after servicing at least an element of the network. This can be expressed rewriting conveniently subtour elimination constraints for a connected graph $G = (C \setminus \{1\}, E)$:

$$\sum_{\forall j:(i,j)\in E(S)} x_{ij} \geq 2, \forall S \subseteq V$$

where $E(S) = \{(i,j) \in E : i \in S, j \in V \setminus S\}$.

We now must extend this inequality to every k-route and taking into account both service and traversing variables:

$$\sum_{\forall j:(i,j)\in E_R^+(S)} x_{ij}^k + \sum_{\forall j:(j,i)\in E_R^-(S)} x_{ji}^k + \sum_{\forall j:(i,j)\in A_R^+(S)} x_{ij}^k + \sum_{\forall j:(j,i)\in A_R^-(S)} x_{ji}^k +$$

$$\sum_{\forall j:(i,j)\in E(S)} y_{ij}^k + \sum_{\forall j:(i,j)\in A(S)} y_{ij}^k \geq 2 \cdot \eta, \forall S \subseteq C, \forall f \in \gamma_R(S), \forall k \in K$$

where

$$\eta = \begin{cases} x_{ij}^k + x_{ji}^k, & \text{if } (i,j) \in E_R \\ x_{ij}^k, & \text{if } (i,j) \in A_R \\ z_i^k, & \text{if } i \in C_R \cap S \end{cases}$$

We introduced this term for limiting subtour elimination constraint to only activated service variable, or to assure every route serves at least a required element. This is a critical class of constraints because number of necessary inequality is equal to

$$K \cdot \sum_{k=2}^{|C|} \binom{|C|}{k}$$

2.3.5 Logical

Our constraints overview is completed writing further inequality that fix priority between z_i^k and x_i^k, y_{ij}^k variables, that is:

$$z_i^k \leq \sum_{j\in V:(i,j)\in E_R^+(i)} x_{ij}^k + \sum_{j\in V:(i,j)\in A_R^+(i)} x_{ij}^k + \sum_{j\in V:(i,j)\in E^+(i)} y_{ij}^k +$$

$$\sum_{j \in V:(i,j) \in A^+(i)} y_{ij}^k \forall k \in K, \forall i \in C_R$$

This means that if we pass with route l for servicing a node h ($z_i^k = 1$), then we need having al least a exiting variable from that node.

2.4 Objective Function.

With the above parameters and variables, a capacitated general routing problem on mixed graph has the objective of minimize the total cost (i.e. traveling distance) of the vehicles for each used route.

We can express this in mathematical form as:

$$\min z^* = \sum_{k \in K} \sum_{(i,j) \in E_R} c_{ij}(x_{ij}^k + x_{ji}^k) + \sum_{k \in K} \sum_{(i,j) \in A_R} c_{ij} x_{ij}^k + \sum_{k \in K} \sum_{(i,j) \in E} c_{ij}(y_{ij}^k + y_{ji}^k) + \sum_{k \in K} \sum_{(i,j) \in A} c_{ij} y_{ij}^k$$

2.5 LP Models for the MCGRP.

Here we present the mathematical formulation of our problem ("complete" model), obtained combining all the constraints we've seen.

2.5.1 Double-Edge variables.

$$\min z^* = \sum_{k \in K} \sum_{(i,j) \in E_R} c_{ij}(x_{ij}^k + x_{ji}^k) + \sum_{k \in K} \sum_{(i,j) \in A_R} c_{ij} x_{ij}^k + \sum_{k \in K} \sum_{(i,j) \in E} c_{ij}(y_{ij}^k + y_{ji}^k) + \sum_{k \in K} \sum_{(i,j) \in A} c_{ij} y_{ij}^k \quad (2.10)$$

$$\sum_{k=1}^{m} (x_{ij}^k + x_{ji}^k) = 1, \forall (i,j) \in E_R \subseteq E \quad (2.11)$$

$$\sum_{k=1}^{m} x_{ij}^{k} = 1, \forall (i,j) \in A_R \subseteq A \tag{2.12}$$

$$\sum_{k=1}^{m} z_{i}^{k} = 1, \forall i \in C_R \tag{2.13}$$

$$\sum_{(i,j)\in E_R} d_{ij}(x_{ij}^{k} + x_{ji}^{k}) + \sum_{(i,j)\in A_R} d_{ij}x_{ij}^{k} + \sum_{i\in C_R} d_i z_i^k \leq Q, \forall k \in K \tag{2.14}$$

$$z_i^k \leq \sum_{j\in V:(i,j)\in E_R^+(i)} x_{ij}^k + \sum_{j\in V:(i,j)\in A_R^+(i)} x_{ij}^k + \sum_{j\in V:(i,j)\in E^+(i)} y_{ij}^k + \sum_{j\in V:(i,j)\in A^+(i)} y_{ij}^k, \quad \forall i \in C_R, \forall k \in K \tag{2.15}$$

$$\sum_{\forall j:\ (i,j)\in A_R^+(i)} x_{ij}^k + \sum_{\forall j:\ (i,j)\in A^+(i)} y_{ij}^k - \sum_{\forall j:\ (j,i)\in A_R^-(i)} x_{ji}^k - \sum_{\forall j:\ (j,i)\in A^-(i)} y_{ji}^k = \sum_{\forall j:\ (j,i)\in E_R^-(i)} x_{ji}^k + \sum_{\forall j:\ (j,i)\in E^-(i)} y_{ji}^k - \sum_{\forall j:\ (i,j)\in E_R^+(i)} x_{ij}^k - \sum_{\forall j:\ (i,j)\in E^+(i)} y_{ij}^k, \quad \forall i \in V, \forall k \in K \tag{2.16}$$

$$\sum_{\forall j:(i,j)\in E_R^+(S)} x_{ij}^k + \sum_{\forall j:(j,i)\in E_R^-(S)} x_{ji}^k + \sum_{\forall j:(i,j)\in A_R^+(S)} x_{ij}^k + \sum_{\forall j:(j,i)\in A_R^-(S)} x_{ji}^k + \sum_{\forall j:(i,j)\in E(S)} y_{ij}^k + \sum_{\forall j:(i,j)\in A(S)} y_{ij}^k \geq 2\cdot\eta, \quad \forall S \subseteq C, \forall f \in \gamma_R(S), \forall k \in K \tag{2.17}$$

$$x_{ij}^k \in \{0,1\}, \forall (i,j) \in E_R \cup A_R, \forall k \in K \tag{2.18}$$

$$y_{ij}^k \in \{0,1\}, \forall (i,j) \in E \cup A, \forall k \in K \tag{2.19}$$

$$z_i^k \in \{0,1\}, \forall i \in C_R, \forall k \in K \tag{2.20}$$

This complete formulation express the problem of minimizing the costs 2.10 over all the activated binary variables (i.e. route variables), under the constraints of assignment (2.11 - 2.13), knapsack (2.14), priority (2.15), parity and balanced-set (2.16) and connection or subtour-elimination 2.17.

In other terms, we need to optimize objective function 2.10, over the constraints that every required edge 2.11 and arc 2.12 is served once, and analogous condition is valid for required nodes 2.13. 2.14 is used for saying, for each vehicle we use the knapsack constraint, whereas 2.15 serves for binding between themselves link and node variables (so called priority constraints).

This last constraint can be more clear thinking i.e. if we pass with first vehicle h time from i node, then we need to go out from i at least h time during route building. 2.15 are parity and balanced set constraints, that assures we want to avoid a route pass through a node without exiting from it: in particular parity assures, roughly speaking, that for each node the number of incoming/outcoming links is always odd (i.e. $2,4,6,\dots$ times); whereas balanced set assures for each node there is, at least, the same number of entering and exiting links. 2.17 are connection inequalities written for a mixed graph, where we defined quantity η as said in 2.3.4.

This formulation has got $|V|+2\cdot|E|+|A|$ variables and a number of constraints equal to:

$$|E_R|+|A_R|+|C_R|+|K|\cdot(1+|C_R|+|V|+\sum k=2\dots|C|\binom{|C|}{k})$$

2.6 Short preliminary computational experiments.

In this section we will show some preliminary experiments we have done for validating and testing our model with double-edge variables. We implemented our model using CPLEX solver and Java 1.6, and ran our test with Intel Duo $T5750$ CPU with 3 GB of RAM.

2.6.1 Instances.

Here we show the first computational experiments with random mixed graph instances varying from 3 to 13 nodes. We assigned capacity $Q = 100$ and varied demands which are distributed uniformly in $[0, \frac{Q}{4}]$, meanwhile costs for every link are uniformly distributed in $[C_{MIN}, C_{MAX}]$ ($C_{MIN} = 1, C_{MAX} = 100$). Nevertheless solving complete formulation CPLEX ends with out-of-memory error, making impossible obtaining an exact solution with complete formulation with $n \geq 10$ nodes instances.

We specify that for skip out problem aimed in section 2.1 with lower-bound, we avoided taking demands value too ”near” to Q: it was seen experimentally that reducing this range aims to solve bigger instances of the same kind.

For sake of simplicity, we now assume $depot \equiv 1$, while other nodes are from 2 to $|N|$: we used a randomized procedure for generating a mixed graph G for running tests, as we describe in follows.

Our procedure could be articulate in two steps:

- generate randomized adjacency matrix $m = [c_{ij}]_{i,j=1,\ldots,|V|}$
- use m for creating a new mixed-graph G with uniformly distributed demands;

In first step we need to give a value for the size n of the matrix; this number will be used as starting input variable for our procedure. Next for each i, j s.t. $1 \leq i < j \leq n$, we assigned a random value to every cost c_{ij} following a normal distribution between $[1, 100]$, considering a real range. Edges and arcs will be equally distributed in graph (i.e. 50%) and, we considered opportunity of having at least:

- an edge $(1, k)$, otherwise
- two arcs $(1, k), (h, 1)$

For the required components, we generate each time a random subset of service arcs, edges and nodes; the fixed capacity is computed as $Q = \frac{D_{MAX}}{2} + 2 \cdot D_{MAX}$, where D_{MAX} is the maximum feasible demand value, fixed a priori (i.e. $D_{MAX} = 18$).

Finally we produce an input file structured as follows: in row 1,2,3,4 we report depot index node, capacity, number of nodes and number of edges. In next $r+4$ rows ($r = 1, \ldots, |E|$) we represent an edge as follows:

```
i j cij dij di dj
```

with obvious meaning of every number, i.e. :

```
2 3 27.0 12 7 0
```

represents edge $(2,3)$ with $c_{ij} = 27$, $d_{ij} = 12$, $d_2 = 7$, $d_3 = 0$. Similarly we represent first the number of arcs and then, in next $r + |E| + 4$ rows ($r = 1, \cdots, |A|$) we report an arc in the same way as edge.

So the input file structure can be summarize as:

```
1
Q
|V|
|E|
i j cij dij di dj
...
|A|
i j cij dij di dj
```

As we said, we consider randomly generated instances from 3 to 13 nodes, and some other instance we've used for a firstly computational test. We represent every mixed graph graphically, showing routes over them only for instances. For the sake of brevity, we will omit draw other routes for avoiding confusion and not really significant representations: nevertheless we report the generated routes ρ_k for each instance that was possible to solve for this particular set of randomly generated ones.

Results are summarized in Table 2.2, representing in every column the following values:

- *id* (instance identifier), here is equal to $|V|$;
- Q, the capacity of every vehicle;
- K, the lower-bound computed as we said in 1;
- D, the sum of all demands;
- $|V|$, the number of nodes;
- $|E|$, the number of edges;
- $|A|$, the number of arcs;
- $|C_R|$, the number of required-nodes;
- $|E_R|$, the number of required-edges;
- $|A_R|$, the number of required-arcs;

2.6.2 Solutions.

In table 2.3 we reported solution we've obtained, representing for every instance needed solving time (in ms) T , z^* value when available, and when not we report OOF for Out Of Memory error .

Every mixed-graph is showed from $e3$ to $e11$ in following figs. ...2.21 - 2.6.2, in which we show for each link (i, j) couple c_{ij}, d_{ij}. Services in route are highlighted in bold on links, and are slanted over required nodes.

Tab. 2.2: instances Features.

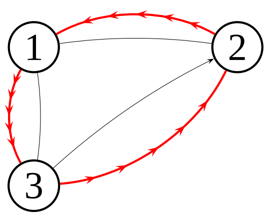
id	**Q**	**K**	**D**	$\|V\|$	$\|E\|$	$\|A\|$	$\|CR\|$	$\|ER\|$	$\|AR\|$
e3	100,00	1	40	3	2	1	1	2	1
e4	100,00	1	31	4	4	2	0	4	2
e5	100,00	2	112	5	7	3	3	7	3
e6	100,00	2	155	6	10	5	4	10	5
e7	100,00	2	129	7	13	8	1	13	8
e8	100,00	3	253	8	13	11	6	17	11
e9	100,00	3	221	9	22	14	3	22	14
e10	100,00	4	336	10	27	18	0	27	18
e11	100,00	4	335	11	32	23	5	32	23
e12	100,00	5	482	12	38	28	7	38	28
e13	100,00	6	588	13	45	33	10	45	33

For each route, we represented in bold the required arcs and edges and in italic required node.

Tab. 2.3: instances Solutions

id	**T[ms]**	**Z***
e3	141,00	131
e4	46,00	422
e5	156,00	461
e6	641,00	860
e7	657,00	1284
e8	6031,00	1618
e9	10031,00	1731
e10	9326296,00	2481
e11	329078,00	2796
e12	OOM	-
e13	OOM	-

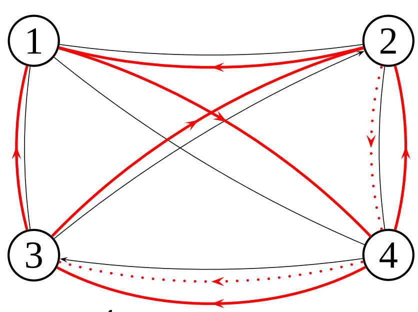
*instance e*3



Instance *e3*

$\rho =$ **(1,3), (3,2), (2,1)** , $c_\rho = 131$

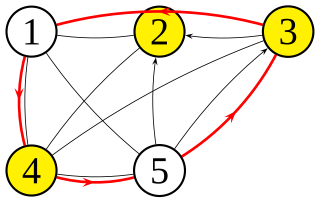
*instance e*4



Instance *e4*

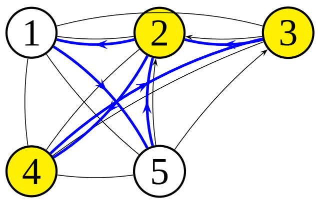
$\rho =$ **(1,4), (4,2), (2,1)**, (1,4), **(4,3), (3,2)**, (2,4), (4,3), **(3,1)** , $c_\rho = 422$

*instance e*5



Instance *e5*

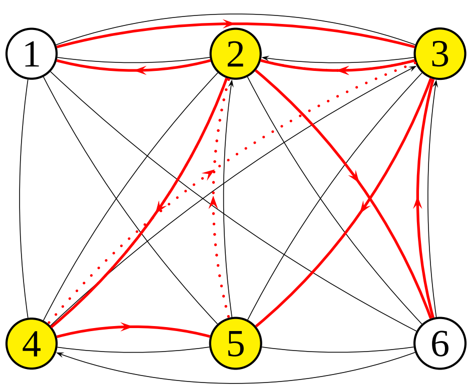
$\rho_1 =$ **(1,*4*), (4,5), (5,*3*), (3,1)** ,$c_{\rho_1} = 150$



Instance *e5*

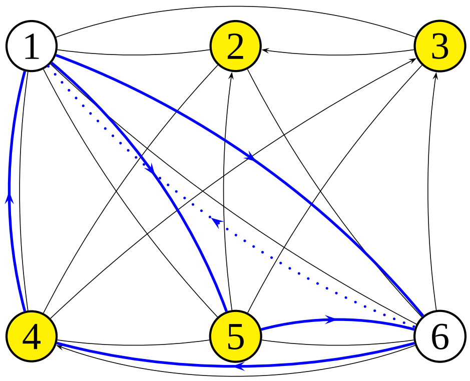
$\rho_2 =$ **(1,5), (5,*2*), (2,4), (4,3), (3,2), (2,1)**, $c_{\rho_2} = 311$

*instance e*6



Instance *e6*

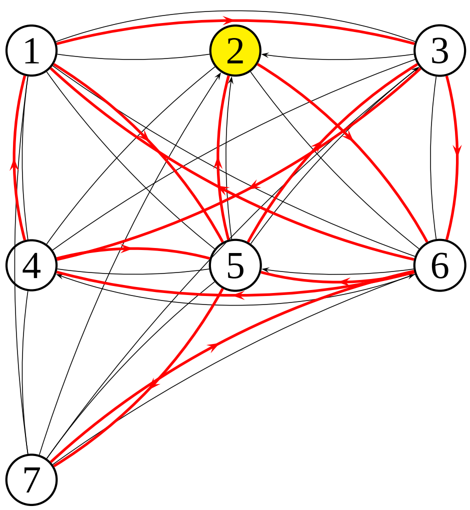
$\rho_1 =$ **(1,3)**, **(3,5)**, (5,*2*), **(2,4)**, (4,*3*), **(3,2)**, (2,4), **(4,5)**, (5,2), **(2,6)**, **(6,3)**, (3,2), **(2,1)**,



Instance *e6*

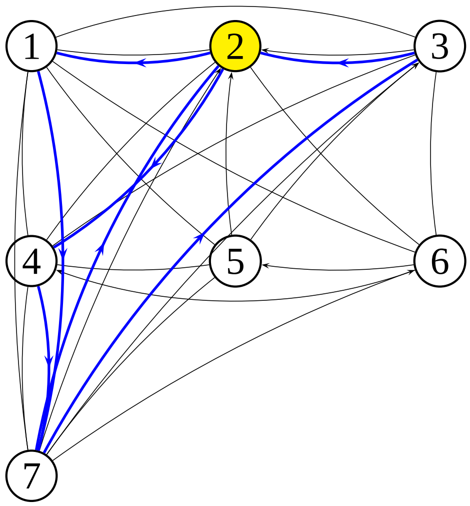
$\rho_2 =$ **(1,6)**, **(6,4)**, **(4,1)**, **(1,5)**, **(5,6)**, (6,1),

instance e7



Instance *e7*

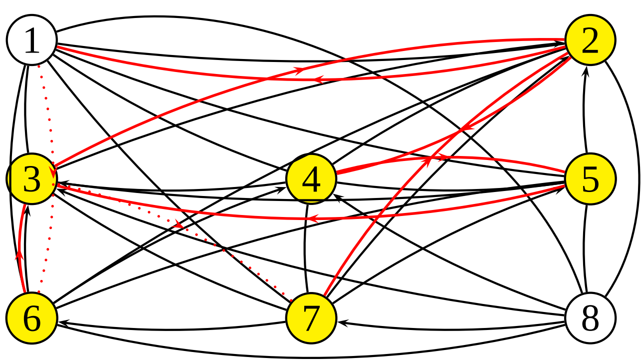
$\rho_1 =$ **(1,3), (3,6), (6,5), (5,3), (3,4), (4,5), (5,7), (7,6), (6,1), (1,5), (5,*2*), (2,6), (6,4), (4,1)**



Instance *e7*

$\rho_2 =$ **(1,7), (7,2), (2,4), (4,7), (7,3), (3,2),(2,1)**

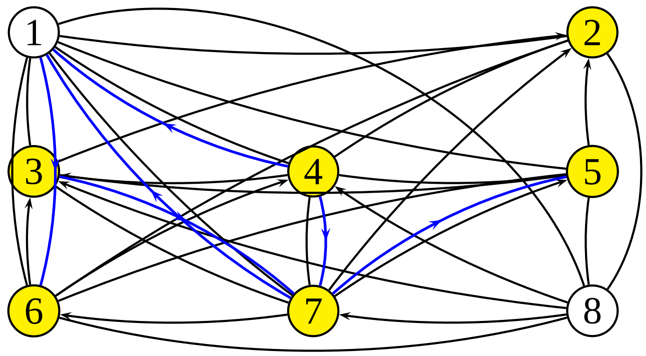
*instance e*8



(2.21)

Instance *e8*

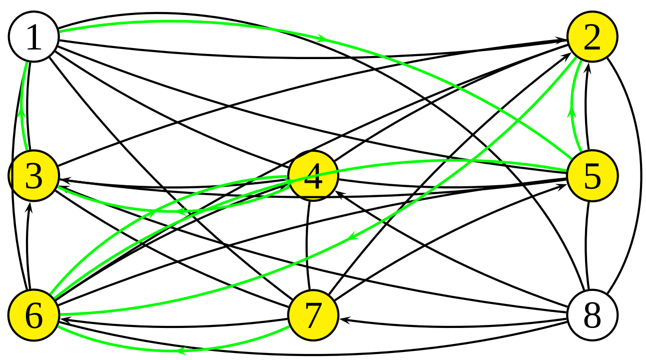
$\rho_1 =$ (1,6), **(6,3)**, (3,7), **(7,2)**, **(2,4)**, **(4,5)**, **(5,3)**, **(3,2)**, **(2,1)**



(2.22)

Instance *e8*

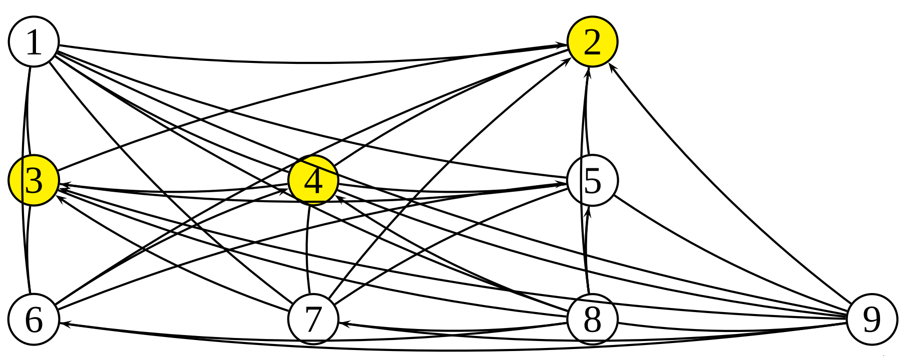
$\rho_2 =$ **(1,*6*)**, **(6,8)**, **(8,3)**,**(3,7)**,**(7,1)**, **(1,8)**, **(8,*4*)**, **(4,7)**,**(7,*5*)**,**(5,8)**, (8,4), **(4,1)**,



(2.23)

Instance *e8*

$\rho_3 =$ **(1,5)**, **(5,2)**, **(2,8)**,**(8,7)**,**(7,6)**, **(6,5)**, (5,2), **(2,6)**,**(6,4)**,**(4,3)**, (3,1)



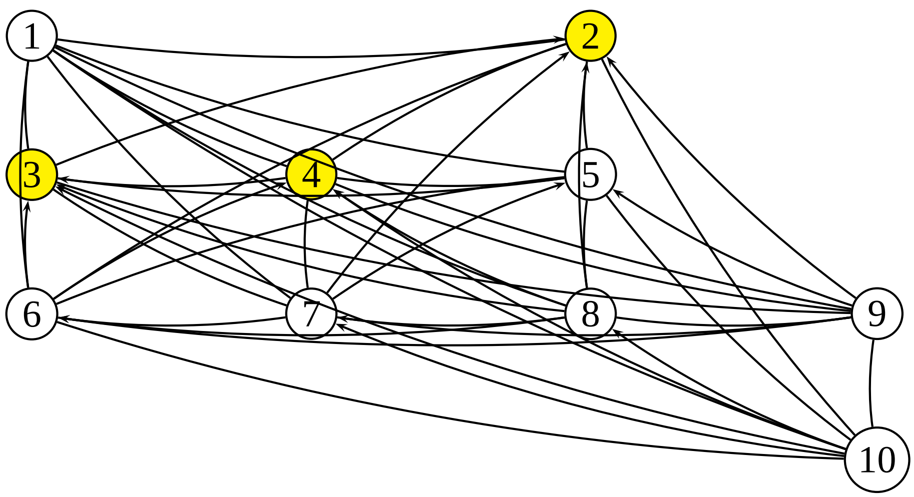
(2.24)

Instance *e9*

$\rho_1 =$ **(1,*4*)**, **(4,3)**, **(3,1)**, **(1,5)**,**(5,2)**, **(2,4)**, **(4,5)**, **(5,7)**, **(7,8)**,**(8,4)**, **(4,7)**, **(7,3)**, **(3,2)**, **(2,1)**

$\rho_2 =$ **(1,9)**, **(9,7)**, **(7,6)**, **(6,4)**,**(4,9)**, **(9,2)**, **(2,8)**, **(8,5)**, **(5,*3*)**,**(3,8)**, **(8,1)**

$\rho_2 =$ **(1,7)**, **(7,*2*)**, **(2,6)**, **(6,5)**,**(5,9)**, **(9,6)**, **(6,8)**, **(8,9)**, **(9,3)**,**(3,6)**, **(6,1)**



(2.25)

Instance *e10*

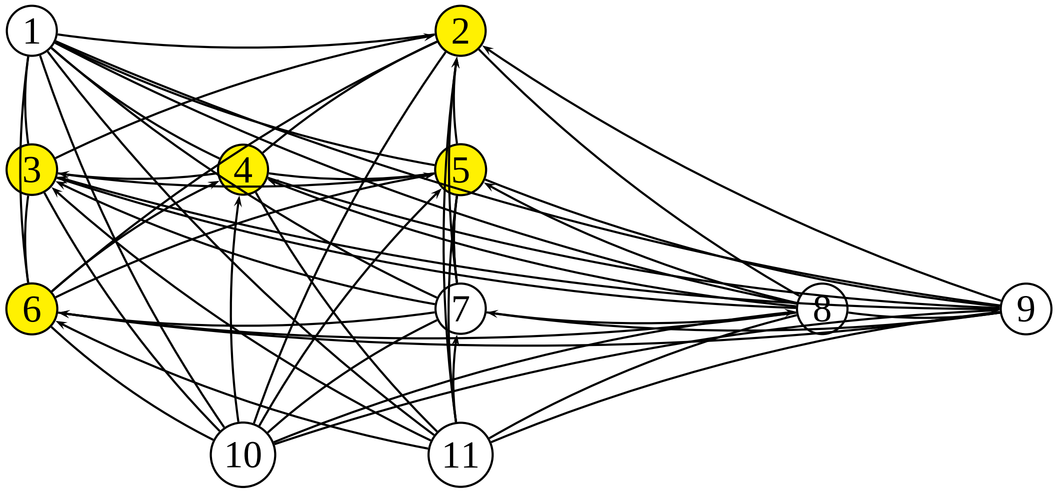
$\rho_1 =$ **(1,5)**, **(5,6)**, **(6,3)**,**(3,9)**, **(9,2)**, **(2,4)**, **(4,1)**, **(1,7)**, **(1,7)**, **(7,5)**, **(5,6)**, **(6,4)**, **(4,5)**, **(5,8)**, **(8,4)**,**(4,1)**

$\rho_2 =$ (1,8), **(8,3)**, **(3,2)**,**(2,6)**, **(6,10)**, **(10,7)**, **(7,9)**, **(9,10)**, **(10,8)**, **(8,7)**, **(7,2)**, **(2,1)**

$\rho_3 =$ **(1,10)**, **(10,3)**, **(3,5)**,**(5,2)**, **(2,10)**, **(10,4)**, **(4,3)**, **(3,7)**, **(7,6)**, **(6,1)**

$\rho_4 =$ **(1,9)**, **(9,6)**, **(6,8)**,**(8,9)**, **(9,5)**, **(5,10)**, **(10,7)**, **(7,4)**, **(4,9)**,

(2,8), **(8,3) (3,1)**



536 Instance *e11*

Part II

UPPER-BOUNDS FOR THE MCGRP.

3. OBTAINING A UPPER-BOUND FOR THE MCGRP.

3.1 Heuristic Algorithm

This algorithm is based over a GRASP (Greedy Randomized Adaptive Search Procedure) approach: in every iteration, it builds up a first feasible solution and then improve it by a local search procedure.

It uses ”*cluster-first, route-second*” approach: in the first phase we try to build a fixed number (m) of cluster C_h, where each one has a certain number of required elements. Matching to each one of them a total demand:

$$D_h = \sum_{(i \in C_R \cap C_h} d_i + \sum_{((i,j) \in E_R \cup A_R \cap C_h} d_{ij}, \forall h = 1, \dots, |C|$$

we must assure that every D_h has the minimum gap with respect to Q. We considered two possible strategies for satisfying this requirement:

Str. 1 Select randomly a seed (required element) for the first cluster C_1, and insert ”nearest” elements $r \in R$ to the one already belonging to C_1, until there are no more residual links or node with compatible demand: repeat same procedure for others cluster, until you’ve finished.

Str. 2 Let m be the number of routes, and define a fictitious capacity $\overline{Q}(j) = \frac{j}{m} \cdot Q, \forall j = 1, \dots, |X|$: now fill cluster j (i.e. there’s at least another compatible element) considering new capacity $\overline{Q}(j)$. In second phase, consider every residual element and insert it into a available cluster, considering capacity Q.

Both of them require a "distance" measurement:

$$\mathfrak{d} : C_j \times t \in C_j \to \mathbb{R}, \forall C_j \in X, \forall t \in C_j$$

that we will specify later in this thesis.

For choosing the best strategy for our purposes, we validated them solving the following model.

$$\min \sum_{k=1}^{m} |\lambda^k - \sum_{s=1, s\neq k}^{m} \lambda^s| \tag{3.1}$$

s.t.

$$\sum_{k\in K} x_{ij}^k = 1, \forall (i,j) \in A_R \tag{3.2}$$

$$\sum_{k\in K} x_{ij}^k + x_{ji}^k = 1, \forall (i,j) \in E_R \tag{3.3}$$

$$\sum_{k\in K} z_i^k = 1, \forall i \in C_R \tag{3.4}$$

$$\sum_{i\in A_R} d_{ij}^k \cdot x_{ij}^k + \sum_{i\in E_R} d_{ij}^k \cdot (x_{ij}^k + x_{ji}^k) + \sum_{i\in C_R} q_i^k \cdot z_i^k \leq Q, \forall k \in K \tag{3.5}$$

$$x_{ij}^k, z_i^k \in \{0,1\}, \lambda^k \in \mathbb{R}_+, \forall k, \forall i \in C_R, \forall (i,j) \in E_R \cup A_R \tag{3.6}$$

This formulation aims to minimize the margin between every cluster lambda and everyone else: in our model that quantity is given by all k-required elements and capacity Q ratio. We then compare the cluster obtained solving this model with the ones obtained by our heuristic procedure, and results seems to confirm his general validity. Obviously we must consider that we ignored the fact that total cost for each cluster could be very high and so very far from the optimum, interesting only to avoiding cluster with demands too much great with respect to others.

For easier solving of model (avoiding absolute value), we introduced constraints:

$$\lambda^k - \sum_{s=1}^{m} \lambda^s = \alpha^k - \beta^k, \forall k \in K$$
$$\alpha^k, \beta^k \geq 0, \forall k \in K$$

and replace objective function 3.1 with:

$$\min \sum_{k \in K} \alpha^k + \beta^k$$

Solving this model for the test-instances seen in previous chapter, it was seen experimentally that the second strategy works better than the first: in fact while the first approach is more fast and produces variable number of cluster (at least m), the second aims to produce a fixed number of cluster m with uniformly distributed demands over all clusters.

For the instance 8e, we have a total demand $D = 253$ so allocated:

- $D(1) = 96, D(2) = 98, D(3) = 59$ for strategy 1;
- $D(1) = 87, D(2) = 66, D(3) = 100$ for strategy 2;

while solving exact model produces:

- $\lambda_1 = 0.84, \lambda_2 = 0.96, \lambda_3 = 0.73$

If we measure:

$$S(i) = \frac{100 \cdot |Q \cdot \lambda_i - D(i)|}{Q \cdot \lambda_i}$$

for $i = 1, 2, 3$ and compute average demand for each strategy, we obtain:

$$\overline{(}S) = (14 + 2 + 14)/3 = 10\%$$

for first strategy and

$$\overline{(}S) = (3.57+31+37)/3 = 23.84\%$$

for the second.

3.2 Metrics: distance definition

A distance over a set $\mathbf{X}$ is a function

$$\delta : \mathbf{X} \times \mathbf{X} \longrightarrow \mathbb{R}$$

which satisfy following properties:

1. $\delta(x,y) \geq 0$
2. $\delta(x,y) = 0 \iff x = y$
3. $\delta(x,y) = \delta(y,x)$
4. $\delta(x,y) \leq \delta(x,z) + \delta(z,y), \forall x,y,z \in \mathbf{X}$

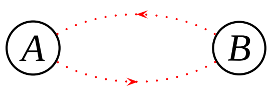
Let G' be an oriented graph obtained from original mixed one G replacing all edges with two opposite arcs and same cost: we build off-line a real matrix $|R| \times |R|$ ($|R| = |C_R| + |E_R| + |A_R|$), in which we compute "mean distances" $d_{ih}, i,h \in \mathbf{X} \equiv R$ as follows. Now let $\delta(R_1,R_2)$ be the shortest path cost between required element couple (R_1,R_2). We distinguish six cases:

- $\delta(A,B) = \frac{d_{AB}+d_{BA}}{2}$, A, $B \in C_R$
- $\delta(AB,C) = \frac{d_{BC}+d_{CA}}{2}$, $AB \in A_R$, $B \in C_R$
- $\delta(AB,CD) = \frac{d_{BC}+d_{DA}}{2}$, AB, $CD \in A_R$
- $\delta(AB,C) = \frac{d_{BC}+d_{CA}+d_{AC}+d_{CB}}{4}$, $AB \in E_R$, $C \in C_R$
- $\delta(AB,CD) = \frac{d_{BC}+d_{CD}+d_{DA}}{3}$, $AB \in A_R$, $CD \in E_R$

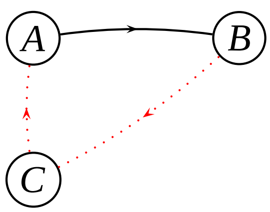
- $\delta(AB,CD) = \frac{d_{BC}+d_{DA}}{2} + \frac{d_{CB}+d_{AD}}{2} + \frac{d_{BD}+d_{CA}}{2} + \frac{d_{DB}+d_{AC}}{2}, AB,CD \in E_R$

Finally we define distance between required element $h \in R$ and a cluster C_j as:

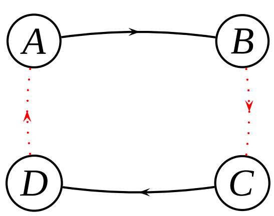
$$\delta(h,C_j) = \frac{1}{|C_j|} \cdot \sum_{i=1}^{|C_j|} d_{ih}, \forall h \in R, \forall C_j \in X$$



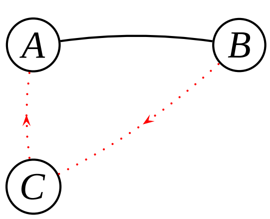
Case 1



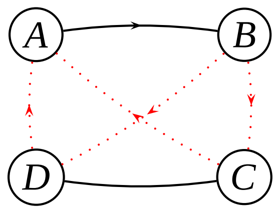
Case 2



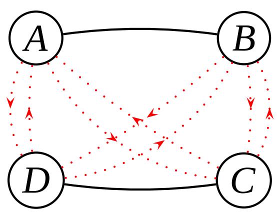
Case 3



Case 4



Case 5



Case 6

3.3 Routing

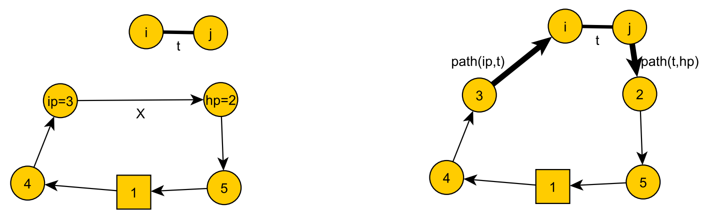
Routing is based over the computing of a *greedy* function $g(t)$:

$$g(t) : (t \in C_j) \longrightarrow \mathbb{R}, \forall C_j \in X$$

where his value is equal to the minimum insertion cost of t in the current route, called ”incremental cost”. We’ve chosen to exploit the simplest (fastest) way for building a route, that is:

- removing minimum cost path between two consecutive nodes (i.e. using notation introduced in 2 ($v_{IP} \equiv v_{\sigma(i)}, v_{HP} \equiv v_{\sigma(i+1)}$, where $IP \neq HP, IP, HP \in V$ stands for respectively insertion point and hook-up point)
- adding $\pi_{IP,t}$ and $\pi_{t,HP}$ (minimum paths between h,t and t,k).

Since we use pre-computed minimum cost paths of the mixed-graph, we're sure that the building route will have a (local) minimum cost. When we build a new route, initially we start with degenerate route $\rho_k = \{depot\}$, and after first insertion of t (either node or link) we will obtain: $\rho_k = \{path(depot,t), path(t,depot)\}$. In general after the $k-$th insertion ($k > 1$) $k-$route will be: $\rho_k = \{\ldots path(IP,t), path(t,HP)\ldots\}$ (in fig. above we showed a t link insertion).



3.4 Algorithm

In what follows we reported the algorithmic outlines of the heuristic.

Algorithm 1 GRASP

Require: Mixed graph G, required elements set $\overline{R} = C_R \cup E_R \cup A_R$, objective function f, greedy function g, parameter $\alpha \in [0,1]$, route set $x = \{\ldots r_k \ldots\}$
Ensure: A feasible solution $\overline{x}$ for MCGRP

  $f(\overline{x}) = \infty$
  for $it = 1$ to $maxiter$ **do**
    $x = \emptyset$
    construct$(G, \overline{R}, g, \alpha)$
    local$(G, \overline{R}, f, x)$
    if $f(x) < f(\overline{x})$ **then**
      $\overline{x} = x$;
      $f(\overline{x}) = f(x)$;
    end if
  end for

Algorithm 2 construct

Require: G, $\bar{R} = C_R \cup E_R \cup A_R$, g, $\alpha \in [0,1]$
Ensure: A feasible solution $\bar{x}$ for MCGRP
  $X \leftarrow$ generateClusters;
  $k = 0$
  while $X \neq \emptyset$ **do**
    $C_j \leftarrow first(X)$
    $r_k = \{depot\}$
    while $C_j \neq \emptyset$ **do**
      $t \leftarrow first(C_j)$
      for all $t \in C_j$ **do**
        compute($g(t)$)
      end for
      $g_{min} = min\{g(t) : t \in C_j\}$
      $g_{max} = max\{g(t) : t \in C_j\}$
      $RCL = \{s \in C_j : g(s) \in [g, g + \alpha(g_{max} - g_{min})]\}$, $\alpha \in [0,1]$
      let $\tilde{s}$ be a random element from RCL set
      $r_k \leftarrow update(r_k, \tilde{s})$
      $C_j \leftarrow C_j \setminus \{\tilde{s}\}$
    end while
    $X \leftarrow X \setminus \{C_j\}$
  end while

Algorithm 3 local

Require: G, $\bar{R}$, f, x
Ensure: A feasible solution $\bar{x}$ for MCGRP
  while $\neg$ localOpt(x) **do**
    $x^{'} = neightboor$(x) such that $f(x^{'}) < f(x)$
    $x = x^{'}$
    $f(x) f(x^{'})$
  end while

This was implemented in Java 1.6 and used for upper-bound computing on all the instances.

4. SOLVING THE MCGRP-LB.

Part III

A BRANCH-AND-CUT ALGORITHM FOR THE MCGRP.

Our branch-and-cut algorithm is based over the checking of violated cut constraints, and subsequent add to model seen in **??**. In what following we introduce three kind of inequalities for our problem, explaining their meaning and including a cutting-plane algorithm for finding and checking them.

5. VALID INEQUALITIES.

5.1 Connectivity Inequalities.

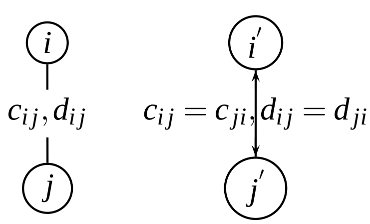
Here we consider the complicating constraints that express connection with depot (2.17):

$$\sum_{(i,j)\in E_R(S)} (x_{ij}^k + x_{ji}^k) + \sum_{(i,j)\in A_R^+(S)} x_{ij}^k + \sum_{(j,i)\in A_R^-(S)} x_{ji}^k + \sum_{(i,j)\in E(S)} (y_{ij}^k + y_{ji}^k) +$$

$$+ \sum_{(i,j)\in A^+(S)} y_{ij}^k + \sum_{(j,i)\in A^-(S)} y_{ji}^k \geq 2\underbrace{(x_{uv}^k + x_{vu}^k)}_{(u,v)\in E_R} \text{ or } 2 \underbrace{x_{uv}^k}_{(u,v)\in A_R} \text{ or } 2 \underbrace{z_s^k}_{s\in S_R};$$

$$\forall S \subseteq V \setminus \{1\}, \gamma_R(S) \neq \emptyset; \forall (u,v) \in E_R(S) \cup A_R(S); \forall s \in S_R; \forall k \in K.$$

These inequalities would be written in exponential number, being $|S|$ the power-set cardinality of all G nodes: clearly this is not done in practice. So we write them checking iteratively only the violated one, adding them to our model and solving the resulting problem; then we will stop procedure when there's no other violation.



5.1.1 Connectivity Inequalities Separation Algorithm.

Let $G^{'} = (V, A^{'})$ be the digraph builded from mixed-graph G replacing every edge with a symmetric couple of two-way arcs (i, j), (j, i), with $c_{ij} = c_{ji}$ and $d_{ij} = d_{ji}$. Let $\mathscr{C}^R = \{C_1^R, \ldots, C_p^R\}$ be strongly R-connected components set of $G^{'}$, and consider $V_{C_1}^R, \ldots, V_{C_p}^R$ as the corresponding vertices set. These components coincide in fact with all the strongly connected subgraphs of $G^{'}$ induced from V_R, $E_R \cup A_R$. Then we write into MCGRP starting formulation the (5.1), for all $S = V_{C_i}^R$ such that $V_{C_i}^R$ doesn't contains depot vertex. Being

$$\bar{S}^k \equiv (\bar{x}^k, \bar{y}^k, \bar{z}^k) \in \mathscr{Z}_+^{2(|E_R|+|E|)+(|A_R|+|A|)+|V_R|}$$

for all $k = 1, \ldots, |K|$ we proceed as follows:

- build graph $\bar{G}^k = (\bar{V}^k, \bar{E}^k, \bar{A}^k)$ in $\bar{G}^k$ where are defined:
  - $\bar{V}^k = \{r \in V | \bar{z}_r^k > 0$ or $\bar{x}_{rj}^k > 0$ or $\bar{y}_{ir}^k > 0$ or $\bar{x}_{rj}^k > 0$ or $\bar{y}_{jr}^k > 0, \forall 1 \leq i \neq j \leq |V|\}$;
  - $\bar{E}^k = \{(h, k) \in E | \bar{x}_{hk}^k > 0$ or $\bar{y}_{hk}^k > 0$ or $\bar{x}_{kh}^k > 0$ or $\bar{y}_{kh}^k > 0$, $\forall 1 \leq i \neq j \leq |V|\}$;
  - $\bar{A}^k = \{(h, k) \in A | \bar{x}_{hk}^k > 0$ or $\bar{y}_{hk}^k > 0, \forall 1 \leq i \neq j \leq |V|\}$;

- determine $G^{'k}$ p connected components (i.e. applying Prim-Dijkstra to every node), and let $\mathscr{C}^{'k} = \{C_1^{'k}, \ldots, C_p^{'k}\}$ be the corresponding vertices set, and $V_{C_1}^{'k}, \ldots, V_{C_p}^{'k}$ their vertices. Between this last set of nodes, remove components with index $1 \leq \bar{p} \leq p$ such that $1 \in V_{C_{\bar{p}}}^{'k}$.

- build an asymmetric support graph $\bar{G}^k = (\bar{V}^k, \bar{E}^k)$ in which consider a fictitious node $s \in \bar{V}^k$ for each connected component with only customers from $G^{'k}$. All of these nodes $s \in \bar{V}^k$ are linked to $t \in \bar{V}^k$ if exists in G at least a link between vertex couple ($V_{C_s}^{'k}$, $V_{C_t}^{'k}$). If no link exists, we insert a fictitious edge, having zero cost, in $\bar{E}^k$. $\bar{E}^k$ is described by:

– edges (s,t) of cost :

$$\sum_{(i,j)\in E_R(V'^k_{C_s}:V'^k_{C_t})}(\overline{x}^k_{ij}+\overline{x}^k_{ji})+\sum_{(i,j)\in E(V'^k_{C_s}:V'^k_{C_t})}(\overline{y}^k_{ij}+\overline{y}^k_{ji})+$$

$$\sum_{(i,j)\in A_R(V'^k_{C_s}:V'^k_{C_t})}\overline{x}^k_{ij}+\sum_{(j,i)\in A_R(V'^k_{C_t}:V'^k_{C_s})}\overline{x}^k_{ji}+$$

$$\sum_{(i,j)\in A(V'^k_{C_s}:V'^k_{C_t})}\overline{y}^k_{ij}+\sum_{(j,i)\in A(V'^k_{C_t}:V'^k_{C_s})}\overline{y}^k_{ji}$$

– $E_R(V'^k_{C_s}:V'^k_{C_t})=\{(i,j)\in E_R : i\in V'^k_{C_s}, j\in V'^k_{C_t}\}$: set of required edges incident into $V'^k_{C_s}$ vertices;

– $E(V'^k_{C_s}:V'^k_{C_t})=\{(i,j)\in E : i\in V'^k_{C_s}, j\in V'^k_{C_t}\}$: set of edges incident into $V'^k_{C_s}$ vertices;

– $A_R(V'^k_{C_s}:V'^k_{C_t})=\{(i,j)\in A_R : i\in V'^k_{C_s}, j\in V'^k_{C_t}\}$: set of required arcs going out from $V'^k_{C_s}$ vertices;

– $A_R(V'^k_{C_t}:V'^k_{C_s})=\{(j,i)\in A_R : j\in V'^k_{C_t}, i\in V'^k_{C_s}\}$: set of required arcs going into $V'^k_{C_s}$ vertices;

– $A(V'^k_{C_s}:V'^k_{C_t})=\{(i,j)\in A : i\in V'^k_{C_s}, j\in V'^k_{C_t}\}$: set of arcs going out from $V'^k_{C_s}$ vertices;

– $A(V'^k_{C_t}:V'^k_{C_s})=\{(j,i)\in A : j\in V'^k_{C_t}, i\in V'^k_{C_s}\}$: set of arcs going into $V'^k_{C_s}$ vertices;

- build the maximum spanning tree (MST^k) over $\overline{G}^k$ (i.e. using Prim-Dijkstra) such that in every step of generation we firstly put a new node $h\in\overline{V}^k$ and then check the violation of inequalities (5.1) in set $V'^k_{C_h}$. If there's a violation, we insert corresponding inequalities in the current problem.

- after building MST, we remove a single edge every time and check inequalities violations into every generated subtree.

In Figure (5.1) we represented a MCGRP instance, with $Q = 10$, and demands/costs are represented by $(c_{ij} \geq 0, d_{ij} \geq 0)$. The optimal solution of mathematical model with assignment, knapsack, priority, parity, balanced-set and connection only for a subset of $R-$connected components is:

- $x^1_{72} = 1; y^1_{17} = y^1_{21} = 1; r_1 = (1-7-2-1); c_1 = 21;$
- $x^2_{39} = 1; y^2_{98} = y^2_{83} = 1; z^2_3 = z^2_8 = 1; r_2 = (3-9-8-3); c_2 = 10;$
- $y^3_{15} = y^3_{51} = 1; z^3_5 = 1; r_3 = (1-5-1); c_3 = 4;$
- $x^4_{14} = x^4_{16} = 1; y^4_{45} = y^4_{51} = y^4_{61} = 1; r_4 = (1-4-5-1-6-1); c_4 = 27;$

where the objective value is $z = 62$.

The connection constraint introduced into starting formulation for $S = \{3, 9\}$ and $k = 2$ is satisfied for the current optimum solution, which does not represent a feasible one for the problem because the following inequality is violated:

$$y^2_{31} + y^2_{32} + y^2_{34} + y^2_{35} + y^2_{82} + y^2_{92} + y^2_{13} + y^2_{23} + y^2_{53} + y^2_{28} + y^2_{29} \geq 2x^2_{39};$$

with $S = \{3, 8, 9\}$. Graph $\overline{G}^2$ is then formed by a unique representative node for the connected component C'^2_1, defined by $V'^2_{C_1} = \{3, 8, 9\}$; so we introduce into current model inequality (**??**).

5.1.2 Algorithmic outline (Connectivity cuts)

In **??** and **??** we reported in pseudo-code the separation algorithm for the connectivity cuts: this will be used in the final part of thesis for computing some significant results. We assumed that d and q

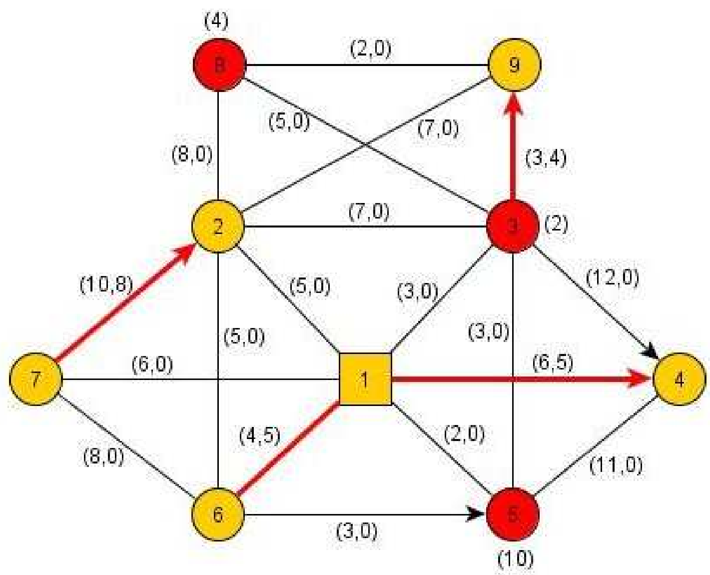


Fig. 5.1: $G = (V, E, A)$.

are the input demand vectors (respectively for links and nodes). The complete separation procedure is also described.

Algorithm 4 connectivitySeparationAlg(K, S')

Require: A feasible solution $S^{'}$ for MCGRP, an integer value K
Ensure: All violated connectivity inequalities with respect to optimal solution of the current relaxation problem (v)

for $k = 1$ to K **do**
  $G'(k) \leftarrow \text{buildMGraph}(G, S', k)$
  if $G'(k) \neq \emptyset$ **then**
    $crs(k) \leftarrow$ connectedComponents(G'(k))
    $crs'(k) \leftarrow$ componentsWithoutDepot(crs)
    if $crs'(k) \neq \emptyset$ **then**
      $\overline{G(k)} \leftarrow \text{buildSupportGraph}(S', crs'(k), k)$
      $\mathbf{v} \leftarrow \mathbf{v} \cup \text{checkAndAddConstraints}(S', \overline{G(k)}, k)$
    end if
  end if
end for
return S';

Algorithm 5 Connection-Cuts

Require: Mixed-graph $G = (V, E, A)$
Ensure: Sub-optimal solution S^*_{CC}

K ← computeLowerBound(d, q, Q)
S' ← solveRelaxedProblem(G,K)
repeat
  $v \leftarrow$ doSepAlg(K, S')
  S' ← updateSolution(v)
until $|v| > 0$

5.2 Co-Circuit Inequalities.

Definition 6: Given a mixed-graph $G = (V, E, A)$ and a node subset $S \subset V$, a link-cutset is defined as the set $\gamma(S) = E(S) \cup A^+(S) \cup A^-(S)$, that is set of all edges and arcs in S nodes.

It is defined for all required links the set $\gamma_R(S) = E_R(S) \cup A_R^+(S) \cup A_R^-(S)$. The co-circuit inequalities assure that every link-cutset being crossed an even number of times, regardless of vehicle being used. Let be $S \subseteq V$, $F \subseteq \gamma_R(S)$ and $F^{'} \subseteq \gamma(S)$, such that $|F| + |F^{'}|$ is odd. The following co-circuit inequalities express the condition that if an odd subset $F \cup F^{'}$ has a vertex into S, then at least an element from $\gamma(S)$ must be served or crossed:

$$\sum_{(i,j)\in\gamma_R(S)\setminus F} x_{ij}^k + \sum_{(i,j)\in\gamma(S)\setminus F^{'}} y_{ij}^k \geq \sum_{(i,j)\in F} x_{ij}^k + \sum_{(i,j)\in F^{'}} y_{ij}^k - |F| - |F^{'}| + 1$$

where $S \subseteq V$, $F \subseteq \gamma_R(S)$, $F^{'} \subseteq \gamma(S)$, $|F| + |F^{'}|$ is odd and $k = 1, \ldots, m$.

In what following we specific every term of this inequality;

- $\sum_{(i,j)\in\gamma_R(S)\setminus F} x_{ij}^k = \sum_{(i,j)\in E_R(S)\setminus F} (x_{ij}^k + x_{ji}^k) + \sum_{(i,j)\in A_R^+(S)\setminus F} x_{ij}^k + \sum_{(j,i)\in A_R^-(S)\setminus F} x_{ji}^k$;

- $\sum_{(i,j)\in\gamma(S)\setminus F'} y^k_{ij} = \sum_{(i,j)\in E(S)\setminus F'} (y^k_{ij} + y^k_{ji}) + \sum_{(i,j)\in A^+(S)\setminus F'} y^k_{ij} + \sum_{(j,i)\in A^-(S)\setminus F'} y^k_{ji}$;
- $\sum_{(i,j)\in F} x^k_{ij} = \sum_{(i,j)\in E_R(S)\cap F} (x^k_{ij} + x^k_{ji}) + \sum_{(i,j)\in A^+_R(S)\cap F} x^k_{ij} + \sum_{(j,i)\in A^-_R(S)\cap F} x^k_{ji}$;
- $\sum_{(i,j)\in F'} y^k_{ij} = \sum_{(i,j)\in E(S)\cap F'} (y^k_{ij} + y^k_{ji}) + \sum_{(i,j)\in A^+(S)\cap F'} y^k_{ij} + \sum_{(j,i)\in A^-(S)\cap F'} y^k_{ji}$.

5.2.1 Co-circuit Inequalities Separation Algorithm.

Cut-Trees.

In the following we will refer to the concepts described in the paper proposed by [1]. Let $G = (V,E)$ be a weighted undirected graph in which a vector of weights $w \in Q^{|E|}_+$ is defined, and let $X \subset V$ be a set of terminal vertices. A cut-tree is an edge-weighted tree spanning X, and representing the minimum cut in G between every pair of vertices in X.

More formally, the cut-tree consists of:

1. a mapping $\pi : V \to T$ such that $\pi(x) = x, \forall x \in X$
2. an adjacency relationship $\sim$, defined on X, such that $x \sim y$ means that x and y are connected by an edge of the tree.

Then if we remove x from a cut-tree, then the set X will be partioned into two disjoint sets X_x and X_y, so that a cut $(U,\overline{U})$ in G (also called ”cut inducted” by edge $x \sim y$) is defined.

Perhaps following condition must hold:

- for every pairs $x,y \in X$ with $x \sim y$, the cut inducted by the edge $x \sim y$ is a minimum $(x,y)-$cut in G with respect to the weights w;

Definition 7: Given a graph G with weights vector w, let H be a connected subgraph of G, and consider a set of vertices $U \subset V$, then the graph which results from G by identification of the vertex set of H as a vertex of U is said supernode. In other words we say that the new graph is obtained by ”shrinking” G.

Given a cut-tree $\mathscr{C}$ defined with respect to T, it satisfy following properties:

1. $\mathscr{C}$ supernodes define a V partition :$V = \bigcup_{S \in \mathscr{L}} S$;
2. Evert vertex of T is exactly contained into a single unique supernode and is said terminal (or representative);
3. let (R,S) be a cut-tree $\mathscr{C}$ branch, and let $r \in T$ e $s \in T$ representatives: (R,S) weight is maximum (r,s)-flow in G: $\lambda_G(r,s) = f(R,S)$;
4. removing (R,S) from $\mathscr{C}$ determine partition of node set in two distinct subsets, which defines a minimum capacity cut in G between r and s, representative respectively for R and S;

Given a supernode R in $\mathscr{C}$, let $(R,S_1),\dots,(R,S_l)$ be branches of tree incident into it:

$$V^{'} := V \backslash U \cup \{u\}, u \notin U; E^{'} := E \backslash (E(U:U) \cup E(U:V \backslash U)) \cup$$
$$\{e = (i,u) | i \notin U, (i,j) \in E, j \in U\};$$
$$A^{'} := A \backslash (A(U:U) \cup A(U:V \backslash U) \cup A(V \backslash U:U)) \cup$$
$$\{a = (i,u) | i \notin U, (i,j) \in A, j \in U\} \cup$$
$$\{a = (u,i) | i \notin U, (j,i) \in A, j \in U\};$$

where $E(U:U)$ $(A(U:U))$ represents edges (arcs) set with extremes into U, while $E(U:V \backslash U)$ $(A(U:V \backslash U))$represents edges (arcs) set with first vertex into U and other into $V \backslash U$, and similarly for $V \backslash U : U$. This operation make possible that U be substituted with a single vertex u in which are concentrated (shrunk) every vertices in U, and then let be removed all parallel links incident into u. So merged link weight is expressed by:

$$\gamma_{iu} = \sum_{\forall j \in U:(i,j) \in E} \gamma_{ij}; \tag{5.1}$$

$$\gamma_{iu} = \sum_{\forall j \in U:(i,j) \in A} \gamma_{ij}; \tag{5.2}$$

$$\gamma_{ui} = \sum_{\forall j \in U:(j,i) \in A} \gamma_{ji}. \tag{5.3}$$

Let $(i, j) \in E \cup A$ be a G link, then graph $G \backslash (i, j)$ is the one we obtain contracting (i, j) through the identification of their vertices ($U = \{i, j\}$): If H is a connected subgraph of G, resulting related graph by shrinking H is equivalent to $U = V(H)$ (by identification of H vertices).

Well-known Gomory-Hu exact algorithm for cut-tree determination is outlined in what following:

Algorithm 6 Cut-tree

Require: Mixed-graph $G = (V, E, A)$ and set $T \subset V$ of terminal vertex.
Ensure: Cut-tree $\mathscr{C}$.

Let be $\mathscr{L} := V$.
while $T \neq \emptyset$ **do**
  Select randomly a $t \in T$ and let be $R \in \mathscr{L}$ supernode in $\mathscr{C}$ where there is t. Let r be R representative.
  Let G_R be shrinking graph obtained by identification of all supernodes $S_1, \dots, S_l$ in $\mathscr{C}$, incident into R, with vertices s_i, $i = 1, \dots, l$.
  Let be $\lambda_{G_R}(r,t) = \lambda_G(r,t)$ max flow from source r to sink t computed over G_R, and let be $\delta(X)$ minimum (r,t)-cut in G_R. Clearly if G_R is disconnected, it is not possible sending flow from r a t, otherwise maximum flow is zero and $\delta(X) = (V_{C_r}, V_{C_t})$, where V_{C_r}, V_{C_t} is respectively the connected components vertices set of r, t.
  Let be $\mathscr{L} = (\mathscr{L} \backslash \{R\}) \cup (\{R \cap X\} \cup (R \cap \overline{X}))$. Supernode R is replaced by supernodes $R \cap X$ and $R \cap \overline{X}$, connected by a link which weight is $f(R \cap X, R \cap \overline{X}) = \lambda_{G_R}(r,t) = \lambda_G(r,t)$.
  $\forall i = 1, \dots, l$, replace every branch (R, S_i) with a new one $(R \cap X, S_i)$ weighted $f(R \cap X, S_i) = f(R, S_i)$ if $s_i \in X$, or a branch $(R \cap \overline{X}, S_i)$ weighted $f(R \cap \overline{X}, S_i) = f(R, S_i)$ if $s_i \in \overline{X}$.
  if $R \cap X$ or $R \cap \overline{X}$ contains only terminal t **then**
    $T = T \backslash \{t\}$.
  end if
end while

Let $G = (V, E, A)$ be the mixed graph:

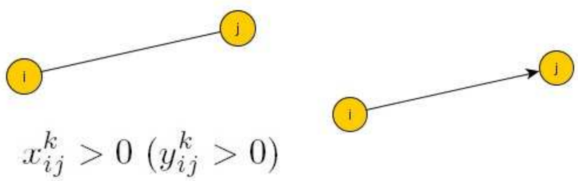
- Let $(\overline{x}, \overline{y}, \overline{z})$ be such that $\overline{x} \in \{0,1\}^{((2|E_R|+|A_R|) \times |K|)}$, $\overline{y} \in \mathscr{Z}_+^{((2|E|+|A|) \times |K|)}$, and let be $\overline{z} \in \{0,1\}^{|C_R| \times |K|}$ relaxed solution. Build related digraph G_k by only variables $\overline{x}_{ij}^k > 0$, $\overline{x}_{ji}^k > 0$, $\overline{y}_{ij}^k > 0$ e $\overline{y}_{ji}^k > 0$.

  From G_k we can define a new related graph G_k^+ as following:

  1. every arc $(i, j) \in G_k$ is splitted into two arcs introducing a new vertex s_{ij} between i and j;
  2. new arc $(i, s_{ij}) \in G_k^+$ is then said *normal half*, and it has even label and capacity $\overline{w}_{is_{ij}}^k = \overline{x}_{ij}^k + \overline{y}_{ij}^k$;

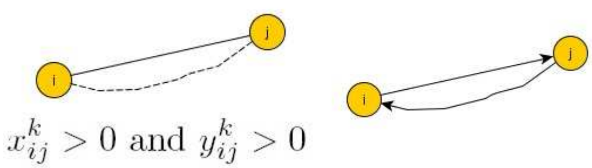
3. complemented arc $(s_{ij}, j) \in G_k^+$ is said *complemented half*, it has odd label and capacity: $\overrightarrow{w}^k_{s_{ij}j} = 1 - \overrightarrow{x}^k_{ij} - \overrightarrow{y}^k_{ij}$.

Every V_k^+ vertex has got even or odd label, if respectively incide a even or odd number of labeled odd arcs: in what following we show typical situation that can occur while building G_k and G_k^+.



(a) Required edge in G ***(b)*** Arc in G_k

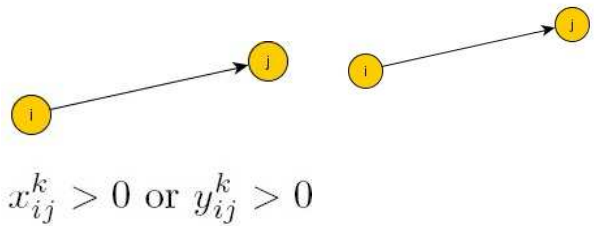
Fig. 5.2: First case in G_k building



(a) Required and dead-headed edge in G ***(b)*** Pair of opposite arcs in G_k

Fig. 5.3: Second case in G_k building

- Let T_k be terminal vertex set defined as odd labeled set in G_k^+;



(a) Arc in G ***(b)*** Arc in G_k

Fig. 5.4: Third case in G_k building

- Invoca l'algoritmo [**6**] su G_k^+ con T_k insieme dei vertici terminali e costruisci il cut-tree $\mathscr{C}_{G_k^+}$.

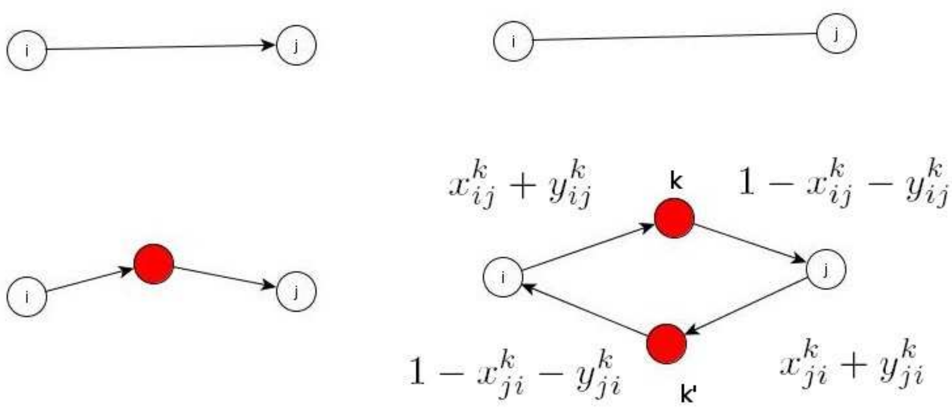


Fig. 5.5: From G_k to G_k^+.

Minimum odd cuts

Let be, without loss of generality, $G = (V, E, \gamma)$ a symmetric weighted graph, with weights $\gamma \in Q_+^{|E|}$ on every edge. Let $T \subset V$ be a node set with even number of odd vertices: a cut $\delta(U)$ is defined T-*odd* (or *odd*) is $|T \cap U|$ is an odd number. The minimum odd cut problem consists in determination of a odd cut $\delta(U)$ having minimum weight $\gamma(\delta(U))$. Padberg & Rao (1982) give a routine for finding this: it firstly call Gomory-Hu procedure for the cut-tree building (with terminal T), and check every branch of the tree for each of the $|T|-1$ cuts which their induce. This algorithm has got complexity equal to $\mathscr{O}(|T||V||E|log(|V|^2/|E|))$.

Padberg-Rao separation algorithm

In what following we report Padberg-Rao algorithm for finding maximum violation of cocircuit inequalities: as a matter of fact, blossom inequalities (originally found by this procedure) is reducible to a minimum odd cut problem

Algorithm 7 Parity cut separation

Require: $G = (V, E, A)$, $S = (\overline{x}, \overline{y}, \overline{z})$

Ensure: Minimum odd sets S_k in which we check co-circuit inequalities violations.

Let be $\varepsilon = 1$.
for $k = 1$ to m **do**
  Let be $S_k = \emptyset$.
  $G_k = RelaxationGraph(G, \overline{x}, \overline{y})$.
  $G_k^+ = AuxiliaryGraph(G_k, \overline{x}, \overline{y})$.
  Determine T_k terminal vertices set (odd nodes in G_k^+): $T_k = GetOdd(G_k^+)$.
  Invoke [**6**] on G_k^+ and build cut-tree $\mathscr{C}_{G_k^+}$: $\mathscr{C}_{G_k^+} = CutTree(G_k^+, T_k)$.
  for each $|T_k| - 1$ branch in $\mathscr{C}_{G_k^+}$ **do**
    Let be $\delta(U_k)$ related cut-set from U_k. Note that U_k is a super-node set of the tree.
    cut-checking: if $|T_k \cap U_k|$ is odd and $\vec{w}^k(\delta(U_k)) = f(U_k : \mathscr{L}_k \backslash U_k) < \varepsilon$ set in S_k original nodes of G such that are contained into U_k supernodes: $S_k = GetVertices(G, U_k)$ for which (**??**) are violated. Note that $f(U_k : \mathscr{L}_k \backslash U_k)$ represents flow on the branch corresponding to $\delta(U_k)$ cut. If there is more than a violation, select minimum cardinality set $U_k^{min} = argmin\{|U_k| : \vec{w}^k(\delta(U_k)) < \varepsilon\}$, and if there exist more minimum sets $S_k = \{S_k^i = GetVertices(G, U_k^i)\}_{i \in M}$, where $M = \{h \in \mathscr{N} : U_k^h = U_k^{min}\}$.
  end for
  for each $S_k^i \in S_k$, let be $F_k^i = \{(i,j) \in \gamma_R(S_k^i) : \vec{x}_{ij}^k > 0 \text{ or } \vec{x}_{ji}^k > 0\}$ e $F_k^{'i} = \{(i,j) \in \gamma(S_k^i) : \vec{y}_{ij}^k > 0 \text{ or} \vec{y}_{ji}^k > 0\}$ cutset for which write the (**??**).
end for

Algorithmic Scheme

For a better performance we select to use the following heuristic: as a matter of fact, our problem is a MIP with integer values and the solution corresponds in every case.

5.3 Balanced-Set Inequalities.

Let be the weights:

$$w_{ij}^k = \overline{x}_{ij}^k + \overline{y}_{ij}^k, \forall (i,j) \in A$$

Algorithm 8 Separation Heuristic for the Co-circuit inequalities

for $i = 1$ to m **do**
  let be $g_k \leftarrow$ related digraph for $x(k) > 0$ or $y(k) > 0$
  for all $n \in N(g_k)$ **do**
    if $isOdd(n)$ **then**
      $\gamma(S) \leftarrow linkCutSet(n)$
      $\gamma_R(S) \leftarrow \gamma(S) \cap (E_R \cup A_R)$
      $F \leftarrow \{(i,j) \in \gamma_R(S)\text{t.c. } \exists x_{ij}^k > 0\}$
      $F' \leftarrow \{(i,j) \in \gamma(S)\text{t.c. } \exists y_{ij}^k > 0\}$
      if $|F| + |F'|$ is odd **then**
        add to the problem violated inequality for n, k
      end if
    end if
  end for
end for

,

$$w_{ij}^k = \overline{x}_{ji}^k + \overline{y}_{ji}^k + \overline{x}_{ij}^k + \overline{y}_{ij}^k, \forall (i,j) \in E$$

825 .

and define $f(S) = w^k(A^+(S)) - w^k(A^-(S)) + w^k(E(S))$. Replacing values we obtain:

$$f(S) = x^k(A_R^+(S)) + y^k(A^+(S)) - x^k(A_R^-(S)) - y^k(A^-(S)) + \\ x^k(E_R(S)) + y^k(E(S)) \geq 0$$

Imposing $f(S)$ not negative means avoiding unbalancing situations, i.e. $c > 0$ ingoing arcs and $a + b < c$ links (a arcs and b edges): so this means that we're imposing that the number of outgoing arcs from S, not balanced from ingoing arcs, must be less or equal to incident edges number. As said in [**?**]:

$$f(S) = w^k(\delta_H(S \cup \{0\}) - P = \sum_{i \in S}(w_i^+ - w_i^-) + w^k(E(S))$$

826 Obviously if $f(S) < 0$ then a violation over current S set is checked.

827 Definition 8: A node set $S \subset V$ having minimum $f(S)$ value is said
828 most unbalanced set.

Norbert & Picard showed in 1996 that this problem is equivalent to determination the maximum of a quadratic function in binary variables opportunely formulated, which for what showed Picard & Ratliff (1975) e Picard & Queyranne (1980) is equivalent solving a maximum flow problem on a related graph with $|V|+2$ nodes.

Let be

$$P=\sum_{i\in V} w_{0i}$$

and consider symmetric graph $H=(V_H,E_H)$ where $V_H=V\cup\{0,n+1\}$ and $E_H=E\cup E_1\cup E_2$, where

$$E_1=\{e=(0,i)\forall i\in V \text{ t.c. } w_e=max\{w_i^- - w_i^+,0\}\}$$

,

$$E_2=\{e=(i,n+1)\forall i\in V \text{ t.c. } w_e=max\{w_i^+ - w_i^-,0\}\}$$

.

Rewriting equation that expresses $f(S)$ we obtain: $w^k(E(S))+\sum_{i\in V\setminus S} w_{0,i}+\sum_{i\in S} w_{i,n+1}-\sum_{i\in V} w_{0i} = w^k(E(S))+\sum_{i\in S}(w_i^+ - w_i^-)$ where we replaced $w_{0,i}=max\{w_i^- - w_i^+,0\}$ and $w_{i,n+1}=max\{w_i^+ - w_i^-,0\}$.

Expressing weights in function of values of current solution variables we have:

$$x^k(E_R(S))+y^k(E(S))+x^k(A_R^+(S))+y^k(A^+(S))-$$
$$x^k(A_R^-(S))-x^k(A^-(S))\geq 0$$

5.3.1 Balanced-Set Separation

- Let be: $w_i^+=w(A+(i))$ and $w_i^-=w(A^-(i)),\forall i\in V$;
- Build capacitated and asymmetric graph $H=(V_H,E_H)$ where $V_H=V\cup\{0,n+1\}$ (0, $n+1$ are fictitious vertices) while $E_H=E\cup E_{0,i}\cup E_{i,n+1}$ (where new sets are double arcs which link 0 and $n+1$ with each other $i\in V$.

Weights corresponds with capacities also defined, and the others are given by:

- $w_{0,i} = max\{w_i^+ - w_i^-, 0\}, \forall i \in V$
- $w_{i,n+1} = max\{w_i^- - w_i^+, 0\}, \forall i \in V$

- Solve a maximum flow problem on H between source $s = 0$ and sink $t = n+1$: minimum capacity cut $S^* \cup \{0\}$ imply that S^* be the most unbalanced set on $\overline{G}^k$.

Please note that in mixed case considering the expression: $f(S) = x^k(A_R^+(S)) + y^k(A^+(S)) - x^k(A_R^-(S)) - y^k(A^-(S)) + x^k(E_R(S)) + y^k(E(S)) \geq 0$

we expressed quantities as following

$$x^k(E_R(S)) = x^k(E_R^+(S)) - x^k(E_R^-(S))$$

,

$$y^k(E(S)) = y^k(E^+(S)) - y^k(E^-(S))$$

,

that is, all (arcs and edges) ingoing contributes are considered with negative sign.

Part IV

RESULTS AND ANALYSIS.

Definition 9: Give a mixed graph $G = (V,E,A)$ with required elements $V_R \subset V$, $A_R \subset A$, $E_R \subset E$, a $R-$connected component of a mixed graph is a mixed subgraph $G' = (V',E',A')$ in which any two nodes $x,y \in V$, $v_1 \neq v_2$ are connected to each other by paths $x = p_1, p_2, \ldots, p_i, p_{i+1}, \ldots, p_l = y$ in which each link p_i, p_{i+1} is such that:

- $i, i+1 \in E_R$;
- $i, i+1 \in A_R$;
- i or $i+1 \in V_R$, $\forall i = 1,2,\ldots,l-1$;

and to which no more nodes or links can be added while preserving its connectivity (maximal connected subgraph).

Every nodes belonging to each distinct G' in G are said $R-$nodes, while the set of all of them will be aimed as **RS**.

Definition 10: Give a mixed graph $G = (V,E,A)$ with required elements $V_R \subset V$, $A_R \subset A$, $E_R \subset E$, a subset R is said $R-$odd iff it has a odd number of inbound and outbound $R-$links.

6.1 A simple relaxed LP Model (MCGRP-LP).

We will show now a simple linear model for obtaining a lower-bound for our problem. Relaxation model is obtained from complete model **??** relaxing constraints 2.17 and rewriting them only for the $R-$nodes just defined. The objective function remains the same as seen previously.

$$\min \ldots \tag{6.1}$$

$$\sum_{k=1}^{m} (x_{ij}^k + x_{ji}^k) = 1, \forall (i,j) \in E_R \subseteq E \tag{6.2}$$

$$\sum_{k=1}^{m} x_{ij}^k = 1, \forall (i,j) \in A_R \subseteq A \tag{6.3}$$

$$\sum_{k=1}^{m} z_i^k = 1, \forall i \in C_R \tag{6.4}$$

$$\sum_{(i,j)\in E_R} d_{ij}(x_{ij}^k + x_{ji}^k) + \sum_{(i,j)\in A_R} d_{ij}x_{ij}^k + \sum_{i\in C_R} d_i z_i^k \leq Q, \forall k \in K \tag{6.5}$$

$$\begin{aligned} z_i^k \leq & \sum_{j\in V:(i,j)\in E_R^+(i)} x_{ij}^k + \sum_{j\in V:(i,j)\in A_R^+(i)} x_{ij}^k + \\ & \sum_{j\in V:(i,j)\in E^+(i)} y_{ij}^k + \sum_{j\in V:(i,j)\in A^+(i)} y_{ij}^k, \\ & \forall i \in C_R, \forall k \in K \end{aligned} \tag{6.6}$$

$$\begin{aligned} & \sum_{\forall j:\ (i,j)\in A_R^+(i)} x_{ij}^k + \sum_{\forall j:\ (i,j)\in A^+(i)} y_{ij}^k - \sum_{\forall j:\ (j,i)\in A_R^-(i)} x_{ji}^k - \\ & \sum_{\forall j:\ (j,i)\in A^-(i)} y_{ji}^k = \sum_{\forall j:\ (j,i)\in E_R^-(i)} x_{ji}^k + \sum_{\forall j:\ (j,i)\in E^-(i)} y_{ji}^k - \\ & \sum_{\forall j:\ (i,j)\in E_R^+(i)} x_{ij}^k - \sum_{\forall j:\ (i,j)\in E^+(i)} y_{ij}^k, \\ & \forall i \in R : R \subset V \text{ is } R-\text{odd}, \forall k \in K \end{aligned} \tag{6.7}$$

$$\sum_{\forall j:(i,j)\in E_R^+(S)} x_{ij}^k + \sum_{\forall j:(j,i)\in E_R^-(S)} x_{ji}^k + \sum_{\forall j:(i,j)\in A_R^+(S)} x_{ij}^k +$$
$$\sum_{\forall j:(j,i)\in A_R^-(S)} x_{ji}^k + \sum_{\forall j:(i,j)\in E(S)} y_{ij}^k + \sum_{\forall j:(i,j)\in A(S)} y_{ij}^k \geq 2\cdot\eta,$$
$$\forall f \in \gamma_R(\ \mathbf{RS}\ ) \forall k \in K \qquad (6.8)$$

Here we summarize the main features of our relaxation:

- report (1)-(6) identically, and solve it at root node;
- write checked-as-violated (8) for every R-connected components;
- write checked-as-violated (7) for every R-odd components;

The resulting value of so builded model will give us z_{LB} value, while z_{UB} was computed with our heuristics fixing iteration number respectively to $maxIter = \ldots$, $maxIteration = \ldots$. Instead computing of z^* value was done following this algorithmic outline, which repeat the procedure adopted for computing z_{LB} until there is at least a violated constraint.

6.1.1 Not-capacitated Instances Results (connectivity-cuts)

We validated our model testing it on some instances used by Corberan et al. for their experimentations on cutting plane algorithm for the General Routing Problem (see **??**). These are not-capacitated instances of mixed graph with demands either over nodes and links, and it is significant because permits to obtain always optimal values with good time performance (only 1 second in such cases). We also note here that instance $GD427$ was not still solved to optimality, and our optimum value $(42550,0)$ is very close to upper-bound (near $0,17\%$) and lower-bound $(0,05\%)$ previously known.

Algorithm 9 3-cuts separation Heuristic

Require: $G = (V, E, A)$, c_{ij}
Ensure: z^*

Solve relaxed model $(1) - (6)$ and let $S = (\overline{x}, \overline{y}, \overline{z})$ be solution.
currViols $\leftarrow \emptyset$
repeat
    size = size(currViols)
    size2 = size(currViols)
    stop $\leftarrow$ updateConstraints(currViols);
    if stop **then**
        break;
    end if
    currViols = currViols $\cup$ parity (currViols)
    currViols = currViols $\cup$ balanced (currViols)
    currViols = currViols $\cup$ connection (currViols)
    size2 = size2 + size(currViols)
until size $\neq$ size2

Name	V	E	A	CR	ER	AR	$\bar{z}$	$\underline{z}$	z^*	USER	CPLEX	ALL	T
alba11	116	158	16	86	14	3	**9419**	**9419**	**9419**	166	18	184	0,1688
alba13	116	125	49	76	17	5	**10744**	**10744**	**10744**	80	14	94	0,5156
alba15	116	99	75	93	7	6	**11332**	**11332**	**11332**	56	3	59	0,0215
alba17	116	96	78	83	11	8	**10795**	**10795**	**10795**	70	13	83	0,0292
alba19	116	77	97	83	11	8	**11410**	**11410**	**11410**	48	4	52	0,0215
alba31	116	160	14	42	45	6	**9870**	**9870**	**9870**	44	53	97	0,0556
alba33	116	126	48	47	35	12	**11315**	**11315**	**11315**	23	23	46	0,0271
alba35	116	108	66	45	32	20	**11435**	**11435**	**11435**	18	28	46	0,0208
alba37	116	90	84	47	26	20	**11742**	**11742**	**11742**	29	12	41	0,0132
alba39	116	89	85	45	28	26	**12766**	**12766**	**12766**	18	21	39	0,0188
alba51	116	157	17	13	81	9	**10931**	**10931**	**10931**	8	57	65	0,2333
alba53	116	126	48	12	65	26	**12480**	**12480**	**12480**	10	24	34	0,0181
alba55	116	103	71	16	51	34	**15558**	**15558**	**15558**	15	31	46	0,0194
alba57	116	102	72	18	55	41	**14893**	**14893**	**14893**	12	18	30	0,0139
alba59	116	104	70	20	58	38	**15848**	**15848**	**15848**	6	38	44	0,0139
alba71	116	161	13	8	116	10	**12566**	**12566**	**12566**	5	120	125	2,0153
alba73	116	119	55	12	81	35	**16647**	**16647**	**16647**	2	60	62	0,0111
alba75	116	106	68	3	83	46	**14887**	**14887**	**14887**	1	54	55	0,0063
alba77	116	97	77	8	71	51	**17427**	**17427**	**17427**	1	52	53	0,0076
alba79	116	84	90	8	59	63	**15501**	**15501**	**15501**	19	30	49	0,0042
alba91	116	164	10	1	148	10	**14497**	**14497**	**14497**	63	104	167	0,1160
alba93	116	138	36	2	124	33	**15680**	**15680**	**15680**	1	107	108	0,0194
alba95	116	98	76	0	88	72	**19032**	**19032**	**19032**	20	28	48	0,0056
alba97	116	87	87	1	76	73	**19338**	**19338**	**19338**	9	16	25	0,0056
alba99	116	90	84	2	79	74	**20026**	**20026**	**20026**	14	26	40	0,0090
GD427	1000	611	1612	292	187	362	42473,9	42574	**42550**	222	64	286	99,3

6.1.2 Capacitated Artificial Instances Results (connectivity-cuts)

Here we tried to solve our artificial instances as done in 2.6, and reported here some results. We considered base dataset seen in first part of the thesis (from 3e to 13e): extending him from 15 links to 59 nodes.

Here we reported previous seen results confronting time needed to close instances: in general we've seen that branch-and-cut is less time-consuming than using the complete formulation. For completeness we reported optimum values in each case (routes was also equivalent).

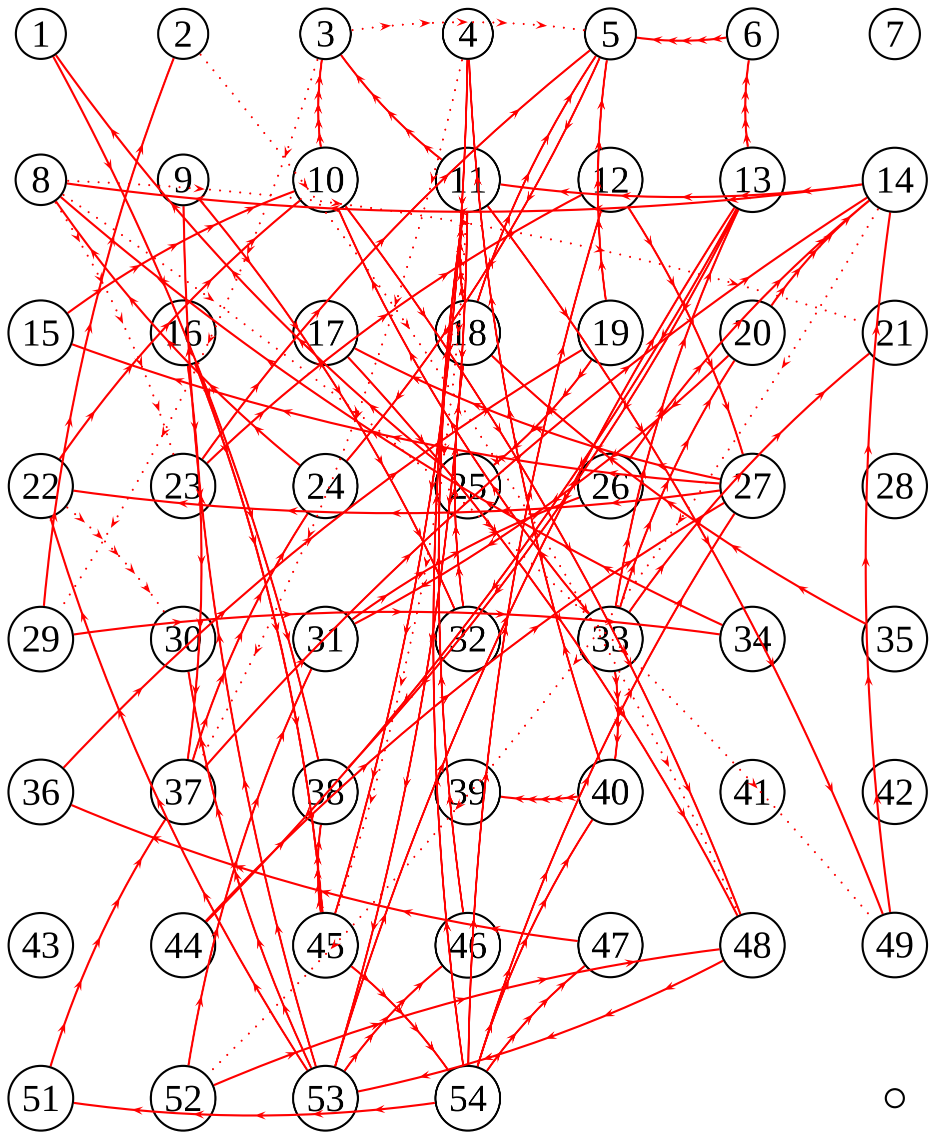
All the instances was closed except for e12, but we note that e13 was instead now closed.

Tab. 6.1: instances Solutions (cuts)

id	**T[ms]**	Z^*	T_{cuts} **[ms]**	Z^*_{cuts}
e3	141,00	131	281,0	131
e4	46,00	422	109,0	422
e5	156,00	461	172,0	461
e6	641,00	860	157,0	860
e7	657,00	1284	218,0	1284
e8	6031,00	1618	1297,0	1618
e9	10031,00	1731	547,0	1731
e10	9326296,00	2481	13421,0	2481
e11	329078,00	2796	1969,0	2796
e12	OOM	-	-	-
e13	OOM	-	62,5	3859

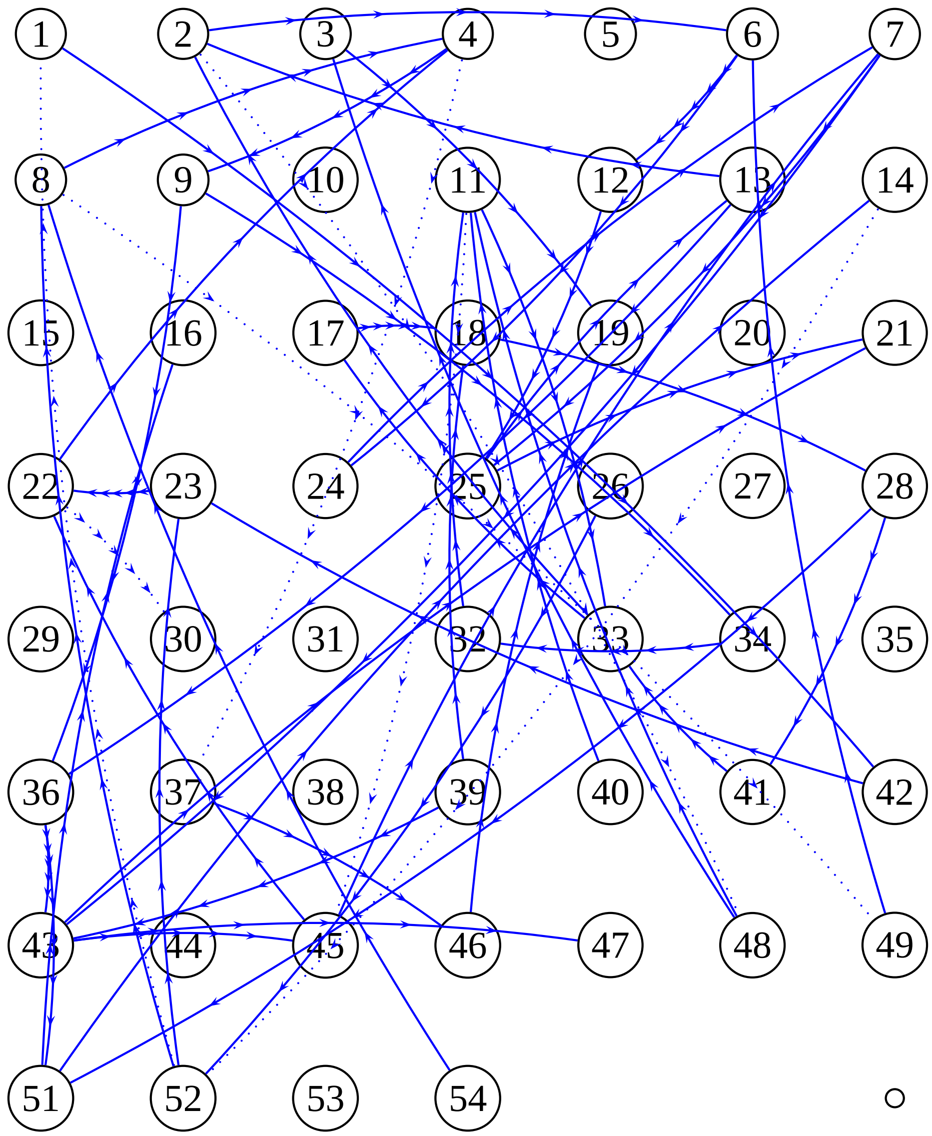
In what following we reported results for another extended set of instances: for some of them was not possible terminating solving procedure for an Out-Of-Memory (OOM) error. We reported here name, K (number of vehicles), V, E, A, CR, ER, AR, number of CPLEX cuts, number of user (connection) cuts, optimum value, seconds required, lower-bound $\underline{z}$ and upper-bound $\overline{z}$ for z.

name	K	V	E	A	CR	ER	AR	CPLEX	USER	CUTS	z^*	Seconds	$\underline{z}$	$\bar{z}$
istanza15e.txt	1	15	59	46	7	5	5	4	1	5	821	0,08	740	1241
istanza18e.txt	2	18	85	68	8	5	6	3	54	57	739	0,32	705	1480
istanza21e.txt	3	21	115	95	8	12	9	-	-	-	OOM	-	1110	2215
istanza24e.txt	3	24	149	127	8	16	7	-	-	-	OOM	-	1232	2736
istanza27e.txt	1	27	188	163	17	13	15	6	0	6	1982	0,13	1878	3334
istanza30e.txt	2	30	232	203	16	27	17	-	-	-	OOM	-	1960	3725
istanza33e.txt	1	33	280	248	21	25	23	18	0	18	2269	0,14	2244	4229
istanza36e.txt	2	36	332	298	23	31	37	0	253	253	3477	8,89	3356	6155
istanza39e.txt	2	39	389	352	16	35	28	-	-	-	OOM	-	3033	5522
istanza42e.txt	1	42	451	410	24	49	35	12	0	12	3888	0,00	3871	7836
istanza45e.txt	2	45	517	473	22	56	43	-	-	-	OOM	-	5033	8127
istanza48e.txt	1	48	587	541	19	50	60	14	0	14	5530	0,20	5520	8901
istanza51e.txt	3	51	662	613	22	56	62	-	-	-	OOM	-	6426	10938
istanza54e.txt	2	54	742	689	35	61	67	0	1748	1748	6958	398,00	6841	11710
istanza57e.txt	2	57	826	770	20	71	88	-	-	-	OOM	-	8114	13655
istanza60e.txt	2	60	914	856	36	99	62	-	-	-	OOM	-	7867	15180



922 Graphical solution for instance *e54*, first route

923



Graphical solution for instance *e54*, second route

7. COMPUTATIONAL COMPLEXITY.

Finally we report our computational complexity analysis either for the GRASP algorithm than the exact approach. We will use the so called $O()$ notation, that is:

Definition 11: an algorithm has time bound $O(f(n))$ if there exist constants N and K such that for every input of size $n \geq N$ the algorithm will not take more than $K \cdot f(n)$ processing time (see **??**).

7.1 GRASP Complexity.

This procedure is made by two parts: in the start we generate clusters, then we try to define a first route over every of them. In the worst case, the shortest path computing for every node in V was computed with Floyd-Warshall algorithm ($O(|V|^3)$), which is the predominant operation with respect to others (metrics, etc.).

7.2 Exact Algorithm Complexity

Complexity analysis was done considering that $S = (x,y,z)$ dimension is equal to $|E_R + A_R| + |E + A| + |C_R|$: in the worst case hypothesis, that is when $E \equiv E_R$, $A \equiv A_R$, $V \equiv C_R$, S cardinality can be expressed as $2(|E + A)|) + |V|$. In our analysis m quantity is considered in our computations, but in typical cases it can be approximated for our purposes as a constant ($m \approx 1$).

RELAXATION. Relaxed model solving requires as predominant action the computing of $R-$connected components int the mixed graph G: this operation was made in our implementation in $O(|V|^2)$, so total complexity is $m \cdot O(|V|^2)$.

PARITY. Parity checking need building m^* digraphs from solution S ($O(1)$), finding odd nodes, computing quantities $\gamma(S), \gamma_R(S), F, F'$ and eventually add a new constraint to the problem. So this procedure has got $m \cdot O(|V|)$ complexity.

BALANCED-SET. This routine, after building support graphs (constant time), requires as predominant action the Ford & Fulkerson algorithm: in general it needs $m \cdot O(|E+A| \cdot f)$. Considering our implementation complexity of this phase is $m \cdot O(|E+A| \cdot |V|^2)$.

CONNECTION. After building support graphs this last phase requires as predominant action the Prim-Dijkstra algorithm $|V|$ times (for computing connected components): using adjacency matrix it needs $m \cdot O(|V|^2)$. Considering our implementation complexity of this phase is $m \cdot O(|V|^3)$.

BIBLIOGRAPHY

[1] D.O. Theis A. N. Letchford, G. Reinelt. A rather long note on the padberg-rao algorithm for capacitated blossom separation. Technical report, 2004.

[2] M. Dror. *Arc routing: theory, solutions, and applications, Vol. 57*. 2002.

[3] J. H. Ramser G. B. Dantzig. The truck dispatching problem. *Management Science*, 6(1):80–91, 1959.

[4] C. S. Orloff. A fundamental problem in vehicle routing. *Networks*, 4(1):35–64, 1974.

[5] B. Muralidharan R. Pandit. A capacitated general routing problem on mixed networks. *Computers and Operations Research*, 22(5):465–478, 1995.

[6] D. Cattrysse D. V. Oudheusden L. Muyldermans, P. Beullens. Exploring variants of 2-opt and 3-opt for the general routing problem. *Operations Research*, 53(6):982995, 2005.

[7] J.M. Sanchis A. Corbern. The general routing problem polyhedron: Facets from the rpp and gtsp polyhedra. *European Journal of Operational Research*, (108):538–542, 1998.

[8] A. Hervas J. C. A. Gutierrez, D. Soler. The capacitated general routing problem on mixed graphs. *Revista Investigacion Operacional*, 23(2):15–16, 2001.

[9] J.M. Sanchis A. Corberan, A. Romero. The mixed general routing problem polyhedron. *Math. Program.*, Ser. A 96(2):103–137, 2003.

[10] J. M. Sanchis A. Corberan, G. Mejia. New results on the mixed general routing problem. *Oper. Research*, 53(2):363–376, 2005.

[11] A. Lodi M. Fischetti. Local branching. *Math. Program.*, Ser. B 98:23–47, 2003.

[12] M.R. Rao M.W. Padberg. Odd minimum cut sets and b-matchings. *Oper. Research*, 7(I):67–80, 1982.

[13] J. M. Sanchis A. Corberan, A. N. Letchford. A cutting plane algorithm for the general routing problem. *Math. Program.*, 7(90):291–313, 2001.

[14] E. Benavent J.M. Belenguer. A cutting plane algorithm for the capacitated arc routing problem. *Comp. & Operations Research*, (30):705–728, 2001.

[15] M. Scantamburlo M. Fischetti, E. Polo. A local branching heuristic for mixed-integer programs with 2-level variables, with an application to a telecommunication network design problem. *Comp. & Operations Research*, 44(Iss. 2):61–72, 2004.

[16] D. O. Theis A. N. Letchford, G. Reinelt. Odd-minimum cut sets and b-matchings revisited. *SIAM Journal on Discrete Mathematics*, 22(Iss. 4):1480–1487, 2004.

[17] Craig A. Tovey. Tutorial on computational complexity. *INFORMS*, 32(3):30–61, 2002.