On the Undirected Rural Postman Problem: Tight Bounds Based on a New Formulation

Elena Fernández[1], Oscar Meza[2], Robert Garfinkel[3], Maruja Ortega[2]

[1] Statistics and Operations Research Department, Technical University of Catalonia, Barcelona, Spain.

[2] Computer Science and Information Technology Department, Simón Bolívar University, Baruta, Venezuela.

[3] School of Business Administration, University of Connecticut, Storrs, Connecticut U.S.A.

e-mail: elena@eio.upc.es, meza@ldc.usb.ve, robg@sba.uconn.edu, mof@usb.ve

ABSTRACT

The Rural Postman Problem (RPP) is a classic "edge-routing" problem. A mathematical programming formulation of RPP that differs fundamentally from those in the literature was introduced, but not tested computationally, by Garfinkel and Webb (1999). A rudimentary algorithm that yields lower bounds via cutting planes and upper bounds via heuristics is developed and tested for a variation of that formulation. Computational results are encouraging, especially in terms of the relatively small number of added inequalities needed to get good lower bounds, and the fact that the vast majority of these have efficient, exact separation procedures. Of particular note is that a first algorithm based on this new formulation is computationally competitive, allowing the possibility of far more efficient and complex future realizations.

1 Introduction

The Rural Postman Problem (RPP) is a classic "edge-routing" problem, as opposed to the Traveling Salesman Problem (TSP), which is "vertex-routing". In particular TSP asks for a minimum length closed walk (tour) on a network that visits a set (generally the entire set) of vertices, while RPP seeks a tour such that a given set of required edges is traversed. Generalizations of RPP include the General Routing Problem (GRP), which allows for required vertices as well as required edges and the Capacitated Arc Routing Problem (CARP), where capacities on the edges may be present. A special case of RPP, for which the subnetwork of required edges is connected, is the Chinese Postman Problem (CPP). Practical applications of edge-routing abound. They include mail delivery, garbage collection, street cleaning, snow plowing, and meter reading and are described in the survey of Eiselt et al. (1995b).

Two equivalent definitions of RPP are the following.

Definition 1

Given: A connected, undirected graph $G = (V, E)$, and edge lengths $d_{ij} \geq 0$ for $\{i, j\} \in E$.

To find: A tour on E, of minimum total length, that traverses each edge in $E_R \subseteq E$ at least once.

Let $E_D = \{\{i, j\} : i \in V,\, j \in V,\, i \neq j\}$.

Definition 2

Given: An undirected graph $G_R = (V, E_R)$, with each component containing at least one edge, and edge lengths $d_{ij} \geq 0$, for $\{i, j\} \in E_D$.

To find: A family of edges E_P of minimum total length, such that $G_{R \cup P} = (V, E_R \cup E_P)$ is Euler (connected with each vertex having even degree) and $\{i, j\} \in E_P$ only if $\{i, j\} \in E_D$. (We do not write $E_P \subseteq E_D$, since E_P may contain multiple copies of edges.)

$G_{R\cup P}$ is Euler and therefore contains a tour that traverses every edge in $E_R \cup E_P$ exactly once. It is easy to see that Definitions 1 and 2 are equivalent. If an edge $\{i, j\}$ of E_D does not exist in E, it can be placed in E_D with length equal to the length of the shortest path between i and j in E. Also, if an edge of E_R is traversed more than once in Definition 1, in Definition 2 it is traversed exactly once as a member of E_R, and the remaining times as a member of E_P. For our purposes Definition 2 is more convenient and we focus on it and on the graph G_R in the rest of the paper.

RPP was shown to be NP-Hard in Lenstra and Rinnooy Kan (1976) and a simpler proof was provided by Garfinkel and Webb (1999). If G_R is connected, however, the resulting CPP is easily solved by low order polynomial algorithms. Thus we will implicitly assume in the rest of the paper that G_R contains at least two components.

Mathematical programming formulations of RPP have been proposed in Christofides et al. (1981), Corberán and Sanchis (1994), and Ghiani and Laporte (1999). Impressive computational results have been achieved by: Ghiani and Laporte (1999) (branch and cut); Corberán et al. (1998) (cutting planes for the GRP); Frederickson (1979) and Hertz et al. (1999) (heuristics).

A mathematical programming formulation that differs fundamentally from the others was introduced, but not tested computationally, by Garfinkel and Webb (1999). Here we modify that formulation to increase its computational viability. We also establish new dominance relations. Finally an algorithm is presented, based on the modified formulation, that yields lower and upper bounds on the optimal solution. Lower bounds are achieved via cutting planes. Upper bounds are provided by a new heuristic that finds good feasible solutions. Computational results are very encouraging. A relatively small number of cuts is generally needed to get very sharp lower bounds. Furthermore, for the vast majority of these cuts fast, exact separation procedures are known. The quality of the heuristic solutions is also very good. Of particular note is that a first, rudimentary algorithm based on this new formulation is computationally competitive, allowing the possibility of far more efficient and

complex future realizations.

Dominance relations and valid inequalities for RPP are discussed in Section 2. Section 3 compares existing mathematical formulations to those based on the work of Garfinkel and Webb (1999). Algorithms in the literature are described briefly in Section 4. The new algorithm, including the proposed heuristic, is the subject of Section 5. Computational results are presented in Section 6 and the last section contains conclusions and future research.

2 Dominance Relations and Valid Inequalities

2.1 An Overview

A number of authors have established *valid inequalities* and *dominance relations* for RPP. Valid inequalities are satisfied by all points of the convex hull of feasible solutions. They are mainly used to eliminate a solution that is infeasible to a problem but feasible to a relaxation of the problem. Typically the constraints relaxed are those calling for integrality.

A dominance relation, on the other hand, takes the objective function into account. Given an optimization problem, a dominance relation is not violated by every optimal solution but may be violated by some. Thus dominance relations can be applied to the original problem to reduce the size of its solution set. This definition of a dominance relation leaves open the possibility that a set of such relations, if imposed together, could eliminate all optimal solutions. For instance if a problem has two optimal solutions, one with $x = 2$ and the other with $x = 3$, then $x \leq 2.5$ and $x \geq 2.5$ would each be a dominance relation but could not be imposed simultaneously. It is easy to verify that the set introduced in this paper does not have this undesirable property, so that any subset can be used without loss of all optimal solutions.

The distinction between dominance relations and valid inequalities becomes blurred if the former are incorporated into a mathematical programming formulation, as is the case in some of the formulations of Section 3. Then it becomes a matter of choice as to which

statement is actually "the problem". Here we will consider Definition 2 to be the generic statement of RPP.

For both valid inequalities and dominance relations two important considerations of their usefulness are depth and efficiency. By depth we mean how much of the feasible region is eliminated by imposing the constraint. In general this is determined computationally although for valid inequalities a surrogate definition is determining whether or not a cut is *facet -inducing*. Efficiency is a separate issue. If, in the context of an LP-based approach for example, a valid inequality or dominance relation is violated by a fractional solution, the question remains as to how easy it is to identify the violated constraint. Thus the distinction is generally made as whether or not there exists a known exact polynomial *separation procedure* that identifies it. If an exact polynomial separation procedure is not known, or sometimes even if it is, there is always the option of using fast heuristic algorithms in search of a valid constraint and resorting to exact algorithms only if the heuristic fails.

Notation: For any set of edges E_Y let $D_Y(i)$, the Y-degree of vertex i, be the number of edges of E_Y incident to i. Vertex i is Y-incident to an edge set E_Y if it is incident to at least one edge in E_Y. Let C_j be the component of G_R containing vertex j. For notational convenience, number the vertices so that vertices 1 through m are in distinct components resulting in C_1,...,C_m being the components of G_R. For any set of vertices $S \subset V$ and edges E_Y, let $\Omega_Y(S)$ be the cutset of S with respect to E_Y. That is, $\Omega_Y(S)$ are the edges in E_Y such that one endpoint of each edge is in S and the other in $V - S$. For any vertex sets $S \subset V$, and $W \subset V$ such that $S \cap W = \phi$, we also write $E_Y(S : W)$ for $\Omega_Y(S) \cap \Omega_Y(W)$, the subset of edges of E_Y with one end point in S and the other in W. The subset of E_Y with both endpoints in S is denoted by $E_Y(S)$. For convenience we often refer to an edge as e rather than by its vertex pair $\{i, j\}$. Let x_e or x_{ij} denote the number of times edge $e = \{i, j\}$ is used in E_P. For any edge set E_Y let $x(E_Y) = \sum_{e \in E_Y} x_e$.

2.2 Dominance Relations

Some of the formulations of RPP introduced in the next section are partially based on dominance relations. Relation 1 is due to Christofides et al. (1981), Relation 2 to Corberán and Sanchis (1994) and Relation 9 to Ghiani and Laporte (1999). The intermediate relations 3 - 8 are from Garfinkel and Webb (1999). Each of the following begins with the phrase "There is an optimal solution E_{P^*} to RPP in which..."

1. ... any edge $\{i, j\}$ is eliminated from E_D, and therefore from E_{P^*} , if $d_{ij} = d_{ik} + d_{kj}$ for some k, where $d_{ik} > 0$ and $d_{kj} > 0$. This relation allows us to assume, for the rest of the paper, that all such edges have been removed from E_D.

2. ...within any component of G_R, any vertex pair i, j is connected by at most one edge of E_{P^*} .

3. ... within any component of G_R, edge $\{i, j\} \in E_{P^*}$ only if $D_R(i)$ and $D_R(j)$ are odd.

4. ... $D_{P^*}(i) \in \{0, 2\}$ if $D_R(i)$ is even.

5. ... $D_{P^*}(i) = 1$ if $D_R(i)$ is odd.

6. ... if $D_R(i)$ is even and there exist j,k, where $k \in C_j$ and $j \neq k$, such that $\{i, j\} \in P^*$ and $\{i, k\} \in P^*$, then $D_R(j)$ and $D_R(k)$ are both odd.

7. ... at most one even R-degree vertex in a given component C_j is P^*-incident to one or more edges in another given component C_k, where $C_j \neq C_k$.

8. ... if $\{i, j\}$ and $\{i, k\}$ are in E_{P^*} where $k \in C_j$ and $C_i \neq C_j$ then $\{i, j\}$ and $\{i, k\}$ are the only edges in E_{P^*} connecting C_i and C_j.

Consider the multi-graph G^c, derived from G_R, by letting the vertices of G^c be the components of G_R and the edges of G^c be those edges of E_D that connect components of G. Edges of G^c retain their distance values from G_R. Let E_{T^*} be the edges of any minimum spanning tree on G^c.

9. ... $x_e \leq 2$, $e \in E_{T^*}$ and $x_e \leq 1$, $e \in E_{P^*} - E_{T^*}$.

Thus at most $m - 1$ edges will be traversed more than once as members of E_{P^*}.

2.3 Valid Inequalities

In the remainder of the paper we will refer to five valid inequalities that have been applied by RPP algorithms. Two of these, namely *Matching* and *Connectivity*, are rather standard and have been applied to other problems. Both have low order exact polynomial separations procedures described by Gusfield (1990) and based on the original work of Gomory and Hu (1961). The three others do not have known exact polynomial separation procedures.

Matching Inequalities

Matching inequalities were introduced by Edmonds (1965) and utilized by Grötschel and Holland (1985) as a means of eliminating "odd cycles" of fractional values for the Perfect Matching Problem. They are:

$$x(E_D(S)) \leq \frac{|S|-1}{2}, \quad S \subset V, \mid S \mid \geq 3 \text{ and odd.} \tag{1}$$

Connectivity Inequalities

Connectivity inequalities were originally proposed by Dantzig, et al. (1954) in order to block subtours for the TSP. They are:

$$x(\Omega_D(S)) \geq 2, \quad S \subset V, \Omega_R(S) = \phi.$$

K-C inequalities

K-C inequalities were first described in Corberán and Sanchis (1994). A K-C configuration is a partition $\{V_0, ..., V_K\}$ of vertex sets of V, with $K \geq 3$, such that:

$V_1, ..., V_{K-1}$ and $V_0 \cup V_K$ are clusters of one or more components of G_R;
$\mid E_R(V_0 : V_K) \mid \geq 2$ and even;
$E_D(V_i : V_{i+1}) \neq \phi$, for $i = 0, ..., K-1$.
The corresponding K-C inequality can be written as

$$(K-2)\cdot x(E_D(V_0:V_k)) + \sum_{\substack{0\le i<j\le K\\ \{i,j\}\neq\{0,K\}}} \mid i-j \mid \cdot\, x(E_D(V_i:V_j)) \ge 2(K-1).$$

Two other valid inequalities are known as Path-bridge and Honeycomb. We do not define them here since they are not used in the algorithm of Section 5. They are used by other algorithms in the literature. Path-bridge inequalities were introduced by Letchford (1997). A separation heuristic for the general case is found in Corberán et al. (1998). Letchford (1997) contains an exact polynomial separation algorithm for a special case. Corberán and Sanchis (1994) is the original reference for Honeycomb inequalities where a separation heuristic is given (see also Corberán et al. (1998) for application to some special cases).

3 Mathematical Programming Formulations

3.1 Importance

In this section we present some extant and new mathematical programming formulations for RPP. Before doing so, we permit ourselves a small digression concerning the question of how to compare them. That is, what are they used for and, in what sense could one say that one is "better" than another. We will use this digression to compare the formulations presented in the rest of this section.

To begin consider three things that mathematical programming formulations can be used for namely: to improve one's intuition about a problem; to establish the computational complexity of a problem; to find solutions to given numerical realizations of the problem. The first and second uses are beyond the realm of this paper but the third is of interest. That is, within the context of using a formulation to solve a realization of a problem, how can one compare different formulations of the same problem? We list three general ways to solve a problem ordered by, in general, decreasing reliance on the formulation.

a. Solve the formulation in its entirety with a (commercial) general purpose mathematical programming package. Here the type (linear, integer, etc.), structure (blocks, SOS

constraints, etc.), and size (number of variables, constraints, etc.) are clearly of paramount importance. They will determine whether or not a given package can be used at all and, if so, whether solution will occur in an acceptable time.

b. Develop one's own algorithm. Typically such algorithms, whether they use: branch-and-bound; branch-and-cut; Lagrangean relaxation, etc. depend intrinsically on the formulation. Relaxations are used to get bounds and these relaxations may be solvable or not solvable or relatively tight or not tight depending on the formulation.

c. Develop heuristics. Typically heuristics are not based directly on formulations although there are certainly exceptions. A typical heuristic, for example a greedy heuristic for a TSP, needs only a statement of the problem, e.g. like Definition 2 for RPP, and not a mathematical programming formulation.

3.2 Formulations of RPP

3.2.1 Formulations in the Literature

An original formulation due to Christofides et al. (1981) was later modified by Corberán and Sanchis (1994) as given below.

Formulation (CS)

$$\text{Min} \sum_{e \in E_D} d_e x_e \tag{2}$$

subject to:

$$x(\Omega_D(v)) \equiv D_R(v) \bmod 2, \quad v \in V \tag{3}$$

$$x(\Omega_D(S)) \geq 2, \quad S \subset V,\ \Omega_R(S) = \phi \tag{4}$$

$$x_e \geq 0 \text{ and integer}, \quad e \in E_D. \tag{5}$$

Constraints (3) establish the Euler property of $E_R \cup E_P$ while connectivity is guaranteed by (4). Note that the conditions of (4) specify that S is a union of vertex sets of the components of G_R. While this formulation has been utilized rather effectively the presence of

both modulo constraints and general (as opposed to binary) integer variables could lead to computational difficulties, especially if solution were by a standard mathematical programming package. CS has $\mid E_D \mid$ general integer variables and number of constraints bounded by $2^m + \mid V \mid$.

A modification of CS that takes advantage of Dominance Relation 9 to convert all variables from general integer to binary is due to Ghiani and Laporte (1999). They first determine a minimum spanning tree E_{T^*} as described in Section 2.2. Then they expand E_D by replacing each of the $m-1$ edges $e \in E_{T^*}$ by two identical copies e' and e'' and letting $x_e = x_{e'} + x_{e''}$.

Formulation GL1

GL1 is identical to CS except that E_D has been expanded by the addition of $m-1$ edges as described above, and (5) is replaced by

$$x_e \text{ binary}, \quad e \in E_D. \tag{5a}$$

Since GL1, but not CS, is based on a dominance relation, all feasible (in terms of Definitions 1 and 2) RPP solutions satisfy the constraints of CS, but only a subset satisfy those of GL1. GL1 eliminates one of the disadvantages of CS, namely the general integer variables. A second formulation in the same paper eliminates the need for modulo constraints. It is

Formulation GL2

$$\text{Min} \sum_{e \in E_D} d_e x_e$$

subject to:

$$x(\Omega_D(v) - F) \geq x(F) - \mid F \mid + 1, \quad v \text{ is } R - odd, \mid F \mid \text{ is even} \tag{6}$$

$$x(\Omega_D(v) - F) \geq x(F) - \mid F \mid + 1, \quad v \text{ is } R - even, \mid F \mid \text{ is odd} \tag{7}$$

$$x(\Omega_D(S)) \geq 2, \quad S \subset V, \Omega_R(S) = \phi$$

$$x_e \text{ binary}, \quad e \in E_D.$$

In (6) and (7) F is any subset of $\Omega_D(v)$. Note that unless $x(F) = \mid F \mid$, (6) and (7) hold

trivially, since the right hand sides are nonpositive. Thus (6) is related, but not equivalent to, what Corberán and Sanchis (1994) call the "R-odd inequality"

$$x(\Omega_D(v)) \geq 1, \quad v \text{ is } R-\text{odd}. \tag{6a}$$

On the other hand it is easy to see that the dominance relations of Section 2.2 could have been used to improve GL2, as presented in Ghiani and Laporte (1999). In particular (6) can be simplified and strengthened and (6a) strengthened by Dominance Relation 5, which says

$$x(\Omega_D(v)) = 1, \quad v \text{ is } R-\text{odd}. \tag{6b}$$

Similarly (7) is called the "R-even inequality" by Ghiani and Laporte (1999). It also follows from Dominance Relation 4 that (7) can be strengthened by

$$x(\Omega_D(v)) \in \{0, 2\}, \quad v \text{ is R-even}. \tag{7a}$$

Constraints (7a) could, in turn, be represented by

$$x(\Omega_D(v)) = 2y_v, \quad v \text{ is R-even and } y_v \in \{0, 1\} \tag{7b}$$

but that would involve the introduction of new binary variables so that it is not clear whether (7) or (7b) is more computationally convenient.

It should also be noted that another minor improvement can be made to both GL1 and GL2 by further imposition of Dominance Relation 5. That is, since only even R-degree vertices can have P^*-degree two, the only edges in E_{T^*} that must have copies added to E_D are those connecting pairs of even R-degree vertices. Thus, in general, the number of variables in both formulations can be reduced.

Formulation (GW)

GW was proposed in (Garfinkel and Webb, 1999) and takes advantage of Dominance Relations 2, 3, 4, and 5. In fact it uses these in a very fundamental way by constructing and operating on a new graph.

Build a new graph $G_A = (V_A, E_A)$ from G. V_A contains all vertices in V plus a new vertex $i^{'}$, located directly on top of every R-even vertex i. Note that $\mid V_A \mid$ is even since the number of odd-degree vertices in any graph is even and since every even-degree vertex is replicated. Thus G_A can potentially admit a perfect matching. E_A contains edges connecting every pair i, j of vertices in V_A, except edges $\{i, j\}$, $\{i^{'}, j\}$, and $\{i^{'}, j^{'}\}$, when j is in the same component C_i as i, $D_R(i)$ is even and $j \neq i$. It follows from the location of the new vertices that distances to vertex $i^{'}$, in G_A are the same, from all other vertices, as distance to vertex i in G. Denote the resulting distance matrix (d^a_{ij}).

Any feasible solution E_P to GW will correspond to a perfect matching E_M in G_A. Thus a copy of every even R-degree vertex i is needed to allow $D_P(i) = 2$. $D_P(i)$ cannot exceed two by Dominance Relation 4. E_P is obtained from E_M as follows: if edge $\{i, j\}$, $\{i, j^{'}\}$, or $\{i^{'}, j^{'}\}$ is in E_M then $\{i, j\}$ is added to E_P. If $\{i, i^{'}\} \in E_M$ then $D_P(i) = 0$. It follows that the resulting $G_{R \cup P}$ is Euler.

Partition the vertices of G_A into $S_A = \{S_1, ..., S_m\}$ where $S_j = C_j \cup \{i^{'} \mid i^{'}$ exists and $i \in C_j\}$ and $C_1, ..., C_m$ are the components of G. Note that $S_1, ..., S_m$ are not components and that there is no idea of a required edge in G_A. Let E^*_A be the set of all edges of E_A whose endpoints are in different elements of S_A. Also let E^+_k be the set of edges incident to vertex k in G_A. Connectivity is assured by introducing directed network flow variables $y_{k\ell}$ (as opposed to the undirected variables $x_{k\ell}$) in a manner similar to those of Finke, et al. (1984) for the Traveling Salesman Problem, and Fischetti, et al. (1993) for the Delivery Man Problem. This flow induces a spanning connected subgraph on the reduced graph of G_A (whose vertices are $S_1, ..., S_m$), in which m units of flow must be sent from S_1 to $S_2, ..., S_m$.

The transformation is illustrated by Figure 1.

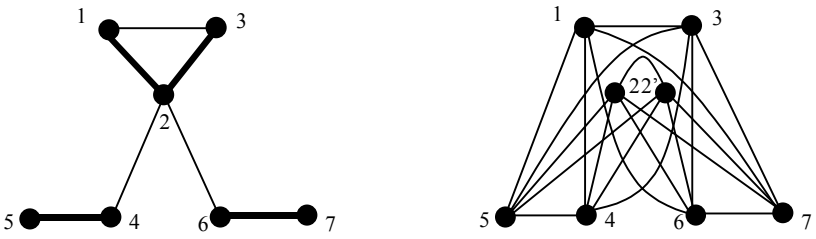

Figure 1

Then the formulation is:

$$\text{minimize} \sum_{\{k,\ell\}\in E_A} d^a_{k\ell} x_{k\ell} \tag{8}$$

subject to:

$$\sum_{\ell:\{k,\ell\}\in E_k^+} x_{k\ell} = 1, \quad k \in V_A \tag{9}$$

$$\sum_{k\in S_1} \sum_{\substack{\ell\notin S_1 \\ \{k,\ell\}\in E_A^*}} y_{k\ell} = m-1 \tag{10}$$

$$\sum_{k\notin S_t} \sum_{\substack{\ell\in S_t \\ \{k,\ell\}\in E_A^*}} (y_{k\ell} - y_{\ell k}) = 1, \quad t = 2,...,m \tag{11}$$

$$y_{k\ell} - (m-1)x_{k\ell} \le 0,\ (k,\ell) = (i,j) \text{ or } (\ell,k) = (i,j),\ \{k,\ell\} \in E_A^* \tag{12}$$

$$x_{k\ell} \in \{0,1\}, \quad \{k,\ell\} \in E_A \tag{13}$$

$$y_{k\ell} \ge 0, \quad (k,\ell) = (i,j) \text{ or } (\ell,k) = (i,j),\ \{k,\ell\} \in E_A^* \tag{14}$$

The objective (8) and constraints (9) and (13) define a minimum cost perfect matching in N_A. The constraints (10), (11), and (14) on the y-variables are those of a simple, single commodity network flow problem. The matching and flow variables are linked in constraints (12). GW has $\mid E_A \mid$ binary and $2 \mid E_A^* \mid - \mid \Omega_A(S_1) \mid$ continuous variables where both $\mid E_A \mid$ and $\mid E_A^* \mid$ can be expected to be somewhat larger than $\mid E_D \mid$ but, depending on the structure of G, may be smaller. It has and $m+ \mid V_A \mid + 2 \mid E_A^* \mid - \mid \Omega_A(S_1) \mid$ constraints, where clearly $\mid V_A \mid$ is somewhat larger than $\mid V \mid$.

3.2.2 A New formulation

A variant of GW that substantially reduces both the number of flow variables and the number of "link" constraints is given below. The flow variables $y_{k\ell}$ connect pairs of sets S_k, S_ℓ rather than node pairs.

Formulation (FMGO)

Replace (10), (11), (12), (14) in GW with:

$$\sum_{k\in\{2,\ldots,m\}} y_{1k} = m-1 \tag{10a}$$

$$\sum_{k\neq t}(y_{kt} - y_{tk}) = 1, \quad t = 2,...,m \tag{11a}$$

$$y_{k\ell} \leq (m-1) \sum_{\substack{\{i,j\}\in E_A^*:\\ \{i\in S_k, j\in S_\ell)\ \cup \\ \{j\in S_k, i\in S_\ell\}}} x_{ij}, \quad k\neq\ell,\ k=1,...,m,\ \ell=2,...,m \tag{12a}$$

$$y_{k\ell} \geq 0, \quad k\neq\ell,\ k=1,...,m,\ \ell=2,...,m. \tag{14a}$$

respectively. FMGO has $\mid E_A \mid$ binary variables, $(m-1)^2$ continuous variables, and $m+\mid V_A \mid + (m-1)^2$ constraints .

3.3 Comparison of formulations

Here we briefly examine the question of how the formulations of Section 3.2 differ from each other. It is clear that the number and types of variables and constraints are not constant among the formulations. A more subtle question is whether or not they are *feasibly equivalent* in the sense of having the same set of feasible solutions based on the original Definitions 1 and 2. That question was partially answered in Section 2.1. To make the argument more precise, let X^{all} be the set of all feasible solutions to RPP as defined by Definition 2. Then let X^{name} be the set of all feasible solutions to any formulation of RPP where *name* is the name of the formulation. Then it should be the case that $X^{name} \subset X^{all}$ if name is based on a dominance relation, and $X^{name} = X^{all}$ otherwise. Furthermore one would expect that two

formulations based on different dominance relations would have different feasible sets. We illustrate those differences for the formulations of Section 3.2.

GL1, GL2, GW, and FMGO are based on dominance relations, while CS is not. In fact $\mid X^{CS} \mid$ is infinite since the modulo constraints do not limit the size of the variables. To see that, for instance, $X^{GL2} \neq X^{FMGO}$, and further that neither set contains the other, consider Figure 2, where the all edges of G are in E_R. GL2 has two feasible solutions each corresponding to a pair of dashed edges in the figure, while FMGO has the unique feasible solution corresponding to $E_P = \phi$.

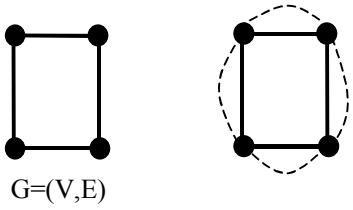

Figure 2

On the other hand Figure 3 shows an FMGO feasible solution x (dashed edges) that does not correspond to any GL2 feasible solution since x_e would have to be at least equal to two in GL2 which is impossible since $e \notin E_{T^*}$ for any E_{T^*}.

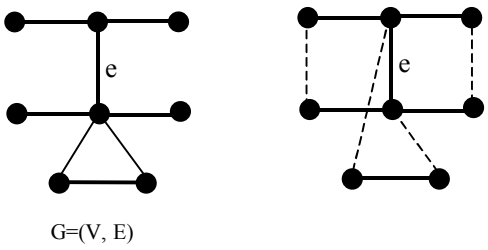

Figure 3

Given that the formulations are different in a fundamental sense it is interesting to look at their aspects that are bound to influence solution techniques. The GW and FMGO formulations are quite different from the others, primarily in that the constraints of CS, GL1, and GL2 use the same x- variables to achieve both the Euler condition and connectivity. GW and FMGO use continuous y-variables for the latter. Of course the two types of variables must be linked as in (12a). On the other hand the possibility of *unlinking* them, perhaps via Lagrangean techniques becomes an attractive option (see the appendix). Finally, because GW and FMGO take advantage of Dominance Relations 2, 3, 4, and 5, the Euler condition can be achieved by obtaining a matching in the augmented network, which is a distinct simplification over the other formulations that impose more general Euler constraints.

4 Algorithms in the Literature

4.1 Cutting Planes

Corberán et al. (1998) present a cutting plane algorithm to get lower bounds for GRP. They solve an LP relaxation based on Formulation CS, with the addition of the constraints $x_e \leq 2$, based on the Dominance Relation 9. When the LP does not terminate integer they add R-odd, K-C, Path-bridge and Honeycomb inequalities and resolve. In general good bounds were achieved although it was often the case that a large number of cutting planes were needed, often of the types for which polynomial separation procedures are not known. Ghiani and Laporte (1999) use a branch-and-cut algorithm based on Formulation GL2 to get exact answers. Upper bounds were found by Frederickson's heuristic. Inequalities used to improve the LP lower bounds were Connectivity, R-odd, and R-even. Separations were solved heuristically despite the existence of polynomial algorithms. Problems of up to 300 vertices were solved in relatively short time, although it was often the case that a large number of inequalities was needed.

4.2 Heuristics

A simple, but effective, heuristic for RPP was proposed by Frederickson (1979). It is a variation of the technique of Christofides (1976) for TSP. The edges of a minimum cost spanning tree E_{T^*}, as described in Section 2.2, are added to G_R, resulting in a connected graph. Then the edges of a minimum cost perfect matching on the odd degree vertices in this new graph are also added yielding a tour. Hertz et al. (1999) have proposed several techniques for improving that solution. The technique is broadly based on the idea of "2-opt" introduced by Lin (1965) for TSP. It uses rather complex construction and post-optimization heuristics.

5 A New Bounding Algorithm

In this section we present a new algorithm to get lower and upper bounds for RPP. Lower bounds are achieved in Phase I by iteratively solving the LP relaxation based on the formulation FMGO, and then adding valid inequalities. Only Connectivity, Matching, and K-C inequalities are used. The first two have efficient separation procedures. K-C inequalities are added only when the other two are not violated. The results of the next section show that, at least for the trial problems, this happens relatively infrequently. Upper bounds are the result of Phase 2, which contains three heuristics, two of which use the output of Phase 1 as input.

5.1 Phase 1 - Lower bounding via cutting planes

The steps of Phase 1 are given below.

1. Let LP be the linear programming relaxation of FMGO, namely (8), (9), (10a), (11a), (12a), (14a) with (13) relaxed to $0 \leq x_{ij} \leq 1$, $\{i, j\} \in E_A$. Go to Step 2.

2. Let x^* be the optimal solution of LP. Go to Step 3

3. If x^* is all integer go to Step 8. Otherwise, go to Step 4.

4. Add Matching and Connectivity inequalities violated by x^* to LP. Go to Step 5.

5. If any inequalities were added in Step 4, go to Step 2. Otherwise search for K-C inequalities violated by x^*. Go to Step 6

6. If any violated K-C inequalities were found in Step 5, add them to LP and go to Step 2. Otherwise, go to Step 7.

7. If the objective function value, $z(x^*)$ is not integer then add $cx \geq \lceil z(x^*) \rceil$to LP, where $\lceil \cdot \rceil$ is "round up". Let x^* be the optimal solution of LP. Go to Step 8.

8. If x^* is integer, stop. E_P given by x^* solves RPP. Otherwise, let $z^- = z(x^*)$ be a lower bound and go to Phase 2.

Explanation

In Step 4 we use the exact algorithms mentioned in Section 2.3 to find violated Matching and Connectivity inequalities. If any exist, at least one and at most $m - 1$, will be found. The heuristic algorithm of Step 6 is similar to the Corberan et al. (1998) technique. It has been adapted to operate on the graph G_A instead of G.

An Example of Phase 1

An example illustrating Phase I is represented by the data of Table 1. It is designed to illustrate most of the steps of the algorithm, including the addition of K-C inequalities in Step 5. The entries of Table 1 are the lengths of the shortest paths between the vertices. An asterisk means that the edge is in G_R. After Step 2 let $G_A(x^*)$ be the graph with vertex set G_A and edge set $\{e \mid x_e^* \neq 0\}$.

****Table 1 here****

Iteration 1. Step 2. x^* is not integer, $z(x^*) = 227.83$ and $G_A(x^*)$ is shown in Figure 4-1.

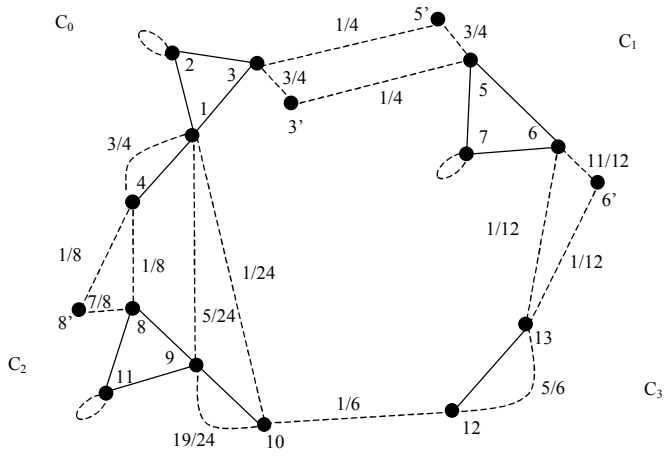

Figure 4-1

Step 4. The three violated Matching inequalities corresponding to: $X = \{4, 8, 8'\}$; $\{6, 6', 10, 12, 13\}$; $\{6, 6', 13\}$ and the three violated Connectivity inequalities corresponding to: C_1; C_2; C_3 are added.

Iteration 2. Step 2. x^* is not integer, $z(x^*) = 389$ and $G_A(x^*)$ is shown in Figure 4-2.

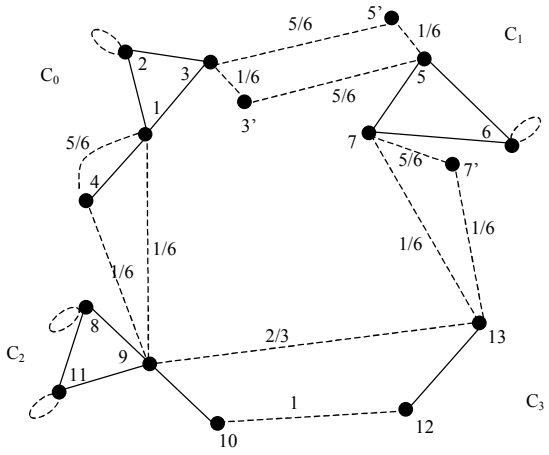

Figure 4-2

Step 4: The violated Matching inequality corresponding to: $X = \{7, 7', 13\}$ and the violated Connectivity inequality corresponding to $C_2 \cup C_3$ are added.

Iteration 3. Step 2. x^* is not integer, $z(x^*) = 426.5$ and $G_A(x^*)$ is shown in Figure 4-3.

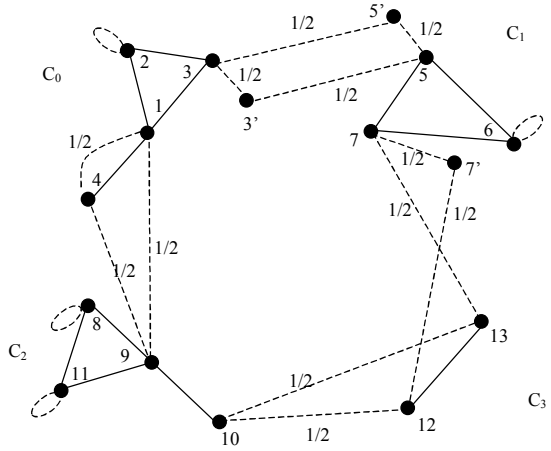

Figure 4-3

Step 4. The violated Matching inequality corresponding to: $X = \{7, 7', 10, 12, 13\}$ is added.

Iteration 4. Step 2. $x^*$is not integer, $z(x^*) = 432.25$ and $G_A(x^*)$ is shown in Figure 4-4.

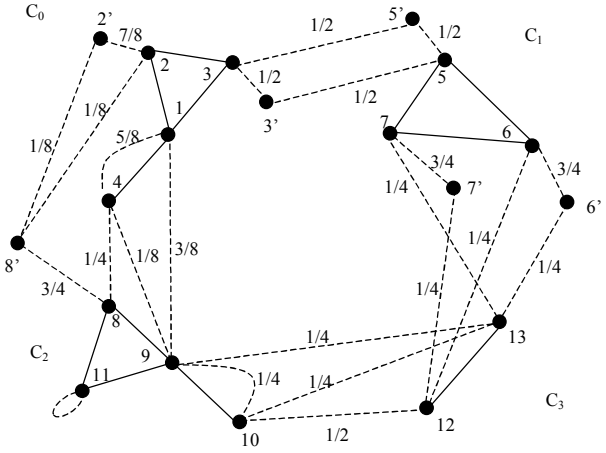

Figure 4-4

Step 4. The three violated Matching inequalities corresponding to: $X = \{2, 2', 4, 8, 8'\}$; $\{2, 2', 8'\}$; $\{6, 6', 7, 7', 10, 12, 13\}$; are added.

Iteration 5. Step 2. x^* is not integer, $z(x^*) = 450.5$ and $G_A(x^*)$ is shown in Figure 4-5.

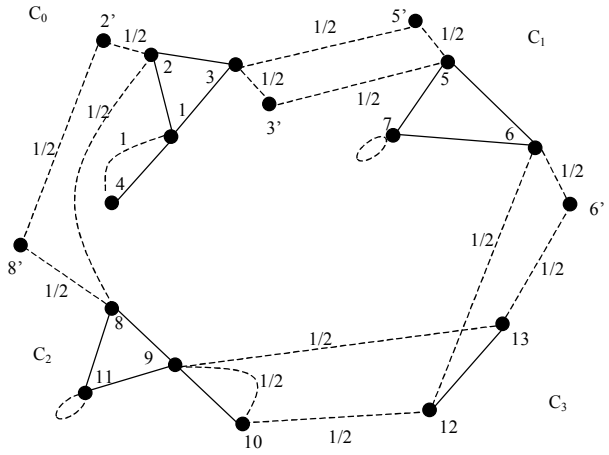

Figure 4-5

Steps 4-6. No violated Matching or Connectivity inequalities exist. The three violated K-C inequalities corresponding to:

$(V_0 = \{2, 2'\},\ V_1 = \{8, 8', 9, 10, 11, 11'\},\ V_2 = \{12, 13\},\ V_3 = \{5, 5', 6, 6', 7, 7'\},\ V_4 = \{1, 3, 3', 4\})$;

$(V_0 = \{5, 5'\},\ V_1 = \{1, 2, 2', 3', 4\},\ V_2 = \{8, 8', 9, 10, 11, 11'\},\ V_3 = \{12, 13\},\ V_4 = \{6, 6', 7, 7'\})$;

$(V_0 = \{8, 8'\},\ V_1 = \{1, 2, 2', 3', 4\},\ V_2 = \{6, 6', 7, 7'\},\ V_3 = \{12, 13\},\ V_4 = \{9, 10, 11, 11'\})$ are added.

Iteration 6. Step 2. x^* is not integer, $z(x^*) = 470$ and $G_A(x^*)$ is shown in Figure 4-6.

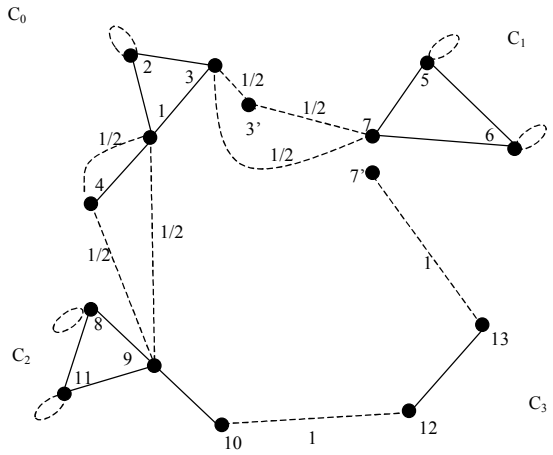

Figure 4-6

Step 4. The violated Matching inequality corresponding to: $X = \{3, 3', 7\}$ is added.

Iteration 7. Step 2. x^* is not integer, $z(x^*) = 470$ and $G_A(x^*)$ is shown in Figure 4-7.

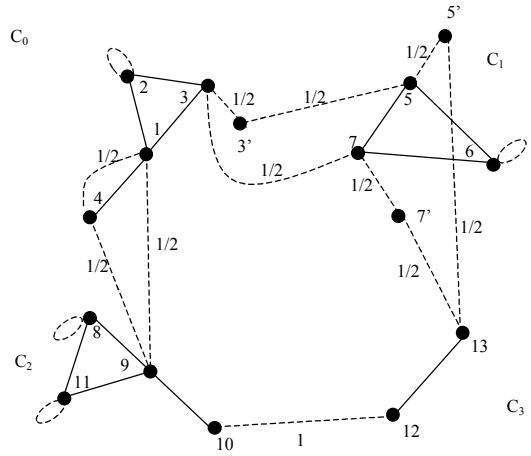

Figure 4-7

Step 4. The violated Matching inequality corresponding to: $X = \{1, 4, 9\}$ is added.

Iteration 8. Steps 2, 3, 8. x^* is integer. $z(x^*) = 473$. The optimal solution is shown in Figure 4-8.

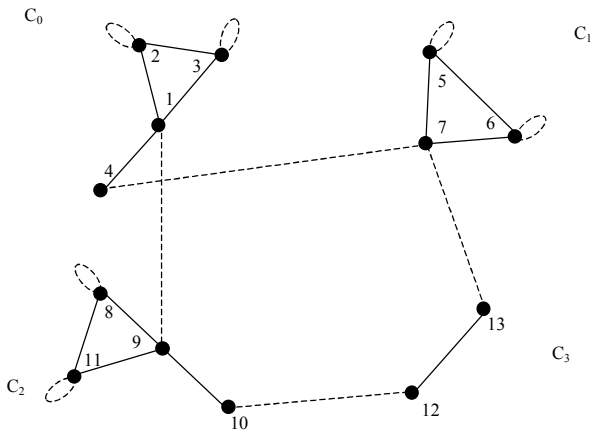

Figure 4-8

5.2 Phase 2 - Upper bounding via the Three Tree Heuristic

The Three Tree Heuristic begins with the data for an RPP and a partial solution $E_T \subseteq E_D$ that corresponds to a tree on the graph G^c defined in Section 2.2. Thus $E_T \cup E_R$ satisfies the connectivity requirement of FMGO but not the Euler requirement. We then solve a CPP taking $E_T \cup E_R$ as the required edges (as in (Frederickson, 1979)) in the original graph. The solution to the CPP is feasible to FMGO. Then, we apply "Euler Reduction" operations (see below) to improve that solution.

We include "Three" in its name only because, in our realization, we experiment with three trees, two of which depend on the output of Phase 1. For those, of course, it is assumed that Phase 1 does not terminate with an all integer solution, since that would be optimal. The Phase 2 algorithm is given below, with explanation following.

The Three Tree Heuristic

Let $z^+ := \infty$, where z^+ is an upper bound on z(RPP).

For $i = 1$ to 3 **do**

Let E_T be the tree E_{Ti} on G^c, and $G_{R \cup E_T} = (V, E_R \cup E_T)$.

Find a minimum distance perfect matching E_M on the odd $(R \cup E_T)$-degree vertices of V. (The resulting solution $E_M \cup E_T$ is feasible to RPP.)

Apply *Euler Reduction* to $E_M \cup E_T$ until no more modification is possible.

Let E_P be the resulting solution and $z^+ = \min\{z^+, z(E_P)\}$.

Endfor

End

As indicated earlier, finding E_M is equivalent to solving a CPP and can be accomplished in $O(n_{odd})^3$ time, where n_{odd} is the number of odd $(R \cup T)$-degree vertices of V(see Grötschel and Holland (1985)).

Euler reduction

The term Euler Reduction of Step 3 was coined in Garfinkel and Webb (1999). For any feasible E_P, it is simply to find any pair of edges that can be deleted or replaced by a single edge without increasing the objective value or losing the Euler property. The steps are:

a. If E_P contains two copies of any edge, and if both can be deleted without disconnecting the graph, do so.

b. If E_P contains edges $\{i, j\}$ and $\{i, k\}$, such that $\{i, j\}$ and $\{i, k\}$ can be replaced by $\{i, k\}$ without disconnecting the graph, do so.

It follows from the nonnegativity of the edge distances and the triangle inequality of shortest paths that the objective cannot increase from either step.

The Three Trees

E_{T_1} is the same tree used by Frederickson (1979) as described in Section 4.2. Trees E_{T_2} and E_{T_3} are designed to extend the Phase 1 solution into a feasible RPP solution as cheaply as possible. All three trees are minimum distance spanning trees on G. They differ in the set of candidate edges, E_{cand} and the distance functions, d_e^*, used to determine them.

E_{T_1}. $E_{cand} = E_D$ and $d_e^* = d_e$.

E_{T_2}. $E_{cand} = \{e \mid x^*_e \neq 0\}$. $d^*_e = 1 - x_e$, with ties broken by d_e. That is, if $x_{e'} = x_{e''}$ and $d_{e'} < d_{e''}$ then $d^*_{e'} < d^*_{e''}$ in the sense that e' is chosen before e'' in the greedy algorithm used to find the minimum distance spanning tree.

E_{T_3}. $E_{cand} = \{e \mid x^*_e \neq 0\}$. $d^*_e = d_e$, with ties broken by x_e.

An example

Phase 2 is illustrated beginning with the fractional solution of Iteration 4, where $z(x^*) = 432.25$. In this case the trees E_{T_1} and E_{T_3} are identical.

$E_{T_1} = \{\{2,8\},\{3,5\},\{10,12\}\}$ and $E_M = \{\{1,3\},\{2,4\},\{5,13\},\{8,9\}\}$. $z(E_{T_1} \cup E_M) = 496$. Three Euler reductions are then applied, none of which changes the objective value. They are: $\{1,3\},\{3,5\}$ replaced by $\{1,5\}$; $\{2,4\}$; $\{2,8\}$ by $\{4,8\}$; and $\{4,8\},\{8,9\}$ by $\{4,9\}$. $z^+ = 496$.

$E_{T_2} = \{\{1,9\},\{3,5\},\{10,12\}\}$ and $E_M = \{\{3,4\},\{5,13\}\}$. $z(E_{T_2} \cup E_M) = 482$. An Euler reduction replaces $\{3,4\}$, $\{3,5\}$ with $\{4,5\}$ reducing the objective to 473. $z^+ = \min\{496, 473\} = 473$.

Thus the final heuristic solution is $\{\{1,9\},\{4,5\},\{5,13\},\{10,12\}\}$ which, as seen from Phase 1, is actually optimal. Note that if we had terminated Phase 1 after Iteration 4 the resulting gap between lower and upper bounds would have been $[433, 473]$.

An observation

As shown by the example, the heuristic of Phase 2 need not be delayed until completion of Phase 1. Since E_{T_2} and E_{T_3} depend on x^* from Phase 1, the heuristics based on these two trees could be run after every iteration of Phase 1 that does not terminate integer. On the other hand E_{T_1} does not depend on Phase 1 so there is no need to run it more than once. In general running the heuristics after every iteration would allow additional feasible solutions to be generated. Then the user might desire to terminate Phase 1 if the resulting gap between lower and upper bounds is sufficiently small.

6 Computational Experience

In this section we report on a series of computational experiments. The benchmark instances were provided to us by Angel Corberán and Alain Hertz based on Corberán et al. (1998) and Hertz et al. (1999). The 158 problems are divided into eight groups. The first group contains two problems, Albaida1 and Albaida 2, obtained from the Albaida, Spain graph (see Corberán and Sanchis, (1994)). Groups 2 and 3 also contain instances derived from the Albaida graph (see Corberán et al. (1998)). The 15 Group 2 problems correspond to RPP instances while the 10 problems of Group 3 are General Routing Problems that are treated as RPP. Group 4 contains 15 instances obtained from the Madriguera, Spain graph (see Corberán et al. (1998)). Group 5 contains the 24 instances of Christofides et al. (1981). Finally, Groups 6, 7 and 8 correspond to instances randomly generated by Hertz et al. (1999). Group 6 contains 20 random graphs. Group 7 consists of 36 grids with disconnected required edge sets, while Group 8 has 36 graphs with vertices of degree four and disconnected required edge sets. In some cases we have divided a group of instances in several subgroups of problems with similar dimension. Table 2 contains information about the type and sizes of the instances.

**** Table 2 here ****

In Corberán et al. (1998), problems in groups G2, G3 and G4 correspond to General Routing Problems. This means that in addition to having required edges, some of the nodes of the graph are also required and have to be visited. In our work we have transformed all the instances in these three groups into RPP instances by eliminating the constraints on the required nodes and only taking into account the required edges.

Programs have been coded in C. All instances, but G2, G3 and G4, were run on an ULTRA SPARC I, with a processor at 143 Mhz, 64bits, and using the CPLEX 6.5 library. The instances in groups G2, G3 and G4 were run on a SUN IPC workstation 10/30 with 4 HyperSparc at 100 using only one processor, and using the CPLEX 4.0 library.

One objective of our experiments is to assess the quality of the model in the context of an

LP-based solution approach. This can be evaluated from two different sources: the quality of the bounds we obtain with our algorithm; and the amount of work required to obtain such bounds. We will first focus on the upper bounds provided by the heuristic of Phase 2. They are compared to those of the Frederickson (1979) heuristic. Since E_{T_2} and E_{T_3} require as input a non-integer solution to the LP relaxation, we only consider those groups of problems for which Phase 1 terminated non-integer.

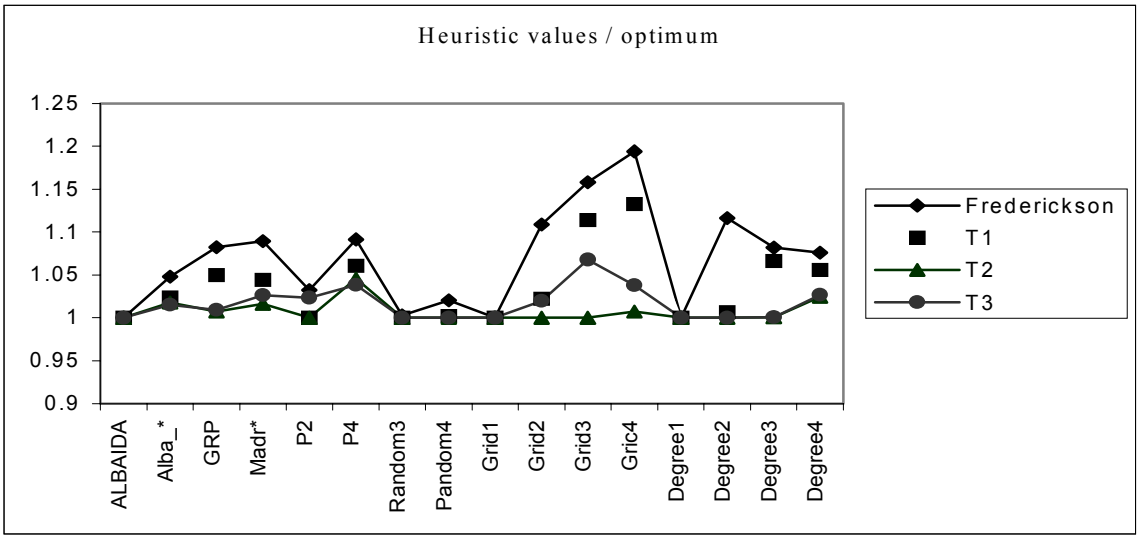

Figure 5. Results of the Three Tree Heuristic

Figure 5 depicts the average ratios between the values obtained with the heuristic and the optimal/best-known values. As was expected E_{T_2} shows only a small improvement as compared to Frederickson. On the other hand, both E_{T_2} and E_{T_3} lead to significant improvements in the quality of the solutions. In general, E_{T_2} seems to have slightly better performance than E_{T_3}. Note that E_{T_2} and E_{T_3} can be applied after any iteration of Phase 1, although this produces a considerable increase of the overall computation time. However, we have observed that when this was attempted the results were generally the same as those obtained when applying E_{T_2} and E_{T_3} only at the end of Phase 1.

Figure 6 compares the results of the Three Tree Heuristic with those of the 2-opt heuristic proposed by Hertz et al. (1999) for the set of randomly generated problems considered in that work. Stacks show the percent deviation from the optimum/best-known solution. They correspond to sets of groups for which at least one of the two approaches did not give the optimal solution for all the instances. When, for one set of problems only one stack is drawn, this means that the other approach generated the optimal solution for all the problems of the group. This happens in the case of Rand1, Grid2, Deg2, and Deg3.

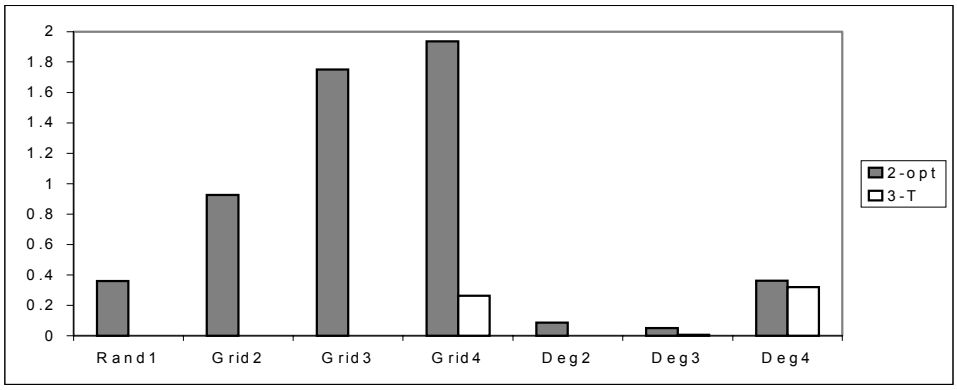

Figure 6. Percent deviation from the optimal/best-known solution

We now focus on the performance of the two-phase algorithm. Table 3 shows a summary of the numerical results. The first two columns contain information relative to the quality of the bounds while the other five reflect the effort required to get the obtained results. In particular, the column *(ub-lb)/lb percent* shows 100*(ub-lb)/lb* where *lb* and *ub* represent, respectively, the values of the lower and upper bounds obtained at termination of the algorithm**.** Column *#opt* represents the number of problems in the corresponding group for which optimality was achieved. In some cases, the optimal solution was obtained in the second phase but optimality could not be proved by comparison with the lower bound. That is, for those problems, verification of the optimality of the Phase 2 solution was achieved

by some additional ad-hoc enumeration. Column *#iterations* shows the number of LP problems solved. Columns *#match*, *#connect* and *#K-C* give, respectively, the number of matching, connectivity and K-C inequalities generated.

****Table 3 here ****

In our opinion these results are very good. On the one hand, 147 out of 158 problems were solved to optimality. For those few for which optimality was not proved, the percentage gaps between the obtained upper and lower bounds are very small. Only for one out of the 20 subgroups of problems (group G4) is the average gap above one percent. On the other hand, these results were obtained rather easily. Notice that both the number of LP iterations as well as the overall number of generated inequalities are small, particularly as compared to other LP-based methods like those of Corberán et al. (1998) and Ghiani and Laporte (1999). For all problems not derived from the Albaida Graph (G4, G5, G6, G7 and G8) the average number of iterations tends to increase with the size of problems and never exceeds 20. A larger number of iterations was required on the average in order to solve the instances from groups based on the real instances extracted from the Albaida graph (groups G1, G2 and G3).

A similar pattern occurs in terms of the number of matching inequalities generated. Again, randomly generated problems (groups G5, G6, G7 and G8) require a small number of such inequalities that tends to increase with the size of problems and real instances (groups G1, G2, G3 and G4) require a larger number of such inequalities. It is not surprising that, in general, the number of generated K-C inequalities is always very small since the algorithm only looks for them when at the current iteration neither matching inequalities nor connectivity inequalities were obtained. This is an important feature of our algorithm that helps to explain the small effort required to obtain the results. As was mentioned in Section 2.3, the types of inequalities widely used (matching and connectivity) can be generated efficiently. However for K-C inequalities no polynomial time separation algorithm is known and we have to resort to heuristic methods (see Corberán et al. (1998)). As can

be seen the number of connectivity inequalities generated during the process is small for all groups and, on the average, never exceeds 30. As could be expected this number has a strong dependency on the number of R-sets. This is illustrated in Figure 7.

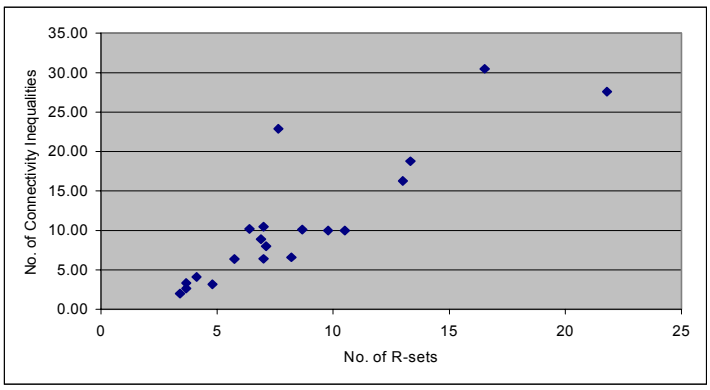

Figure 7. Number of connectivity inequalities vs. number of R-sets

The next figures illustrate the small number of inequalities required by our algorithm by comparing them with the average number of inequalities required by a similar approach like the one of Corberán et al. (1998). Figure 8a depicts the average number of generated connectivity inequalities in Corberán et al. (1998) and in our approach for the groups of problems that were solved as RPP instances in that work. Figure 8b shows the same information but now relative to the average number of other types of inequalities generated during the process. In the two cases for each group of problems the left stack corresponds to Corberán et al. (1998) while the right stack corresponds to our approach. It should be noticed that the scale of the y-axis is different in the two figures. In Figure 8a it ranges from 0 to 25, in Figure 8b its maximum value is 450.

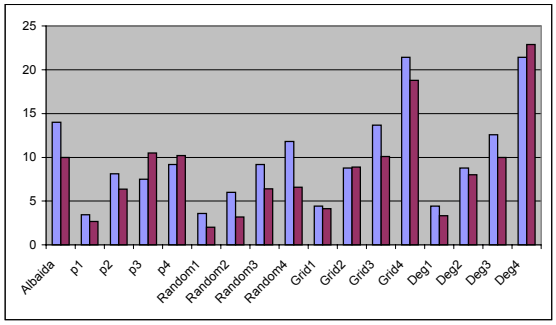

Figure 8a. No. of Connectivity Inequalities

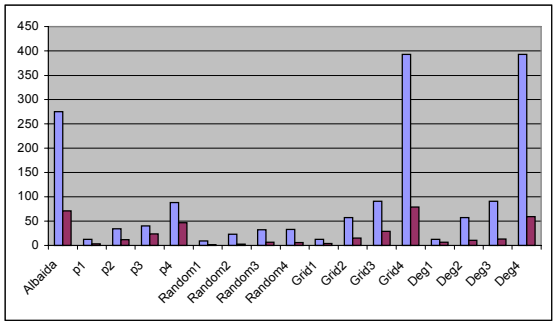

Figure 8b: No. of other types of inequalities

The average times required by our algorithm are given in Figures 9. As can be seen, in general the times are very small. The exception are problems in group G4 (Madr*) which, on average, required about 500 cpu seconds. Thus, problems in this group have not been included in the figure, so as to make it more informative. As can be seen, for the rest of the groups, times are quite small for problems of these sizes.

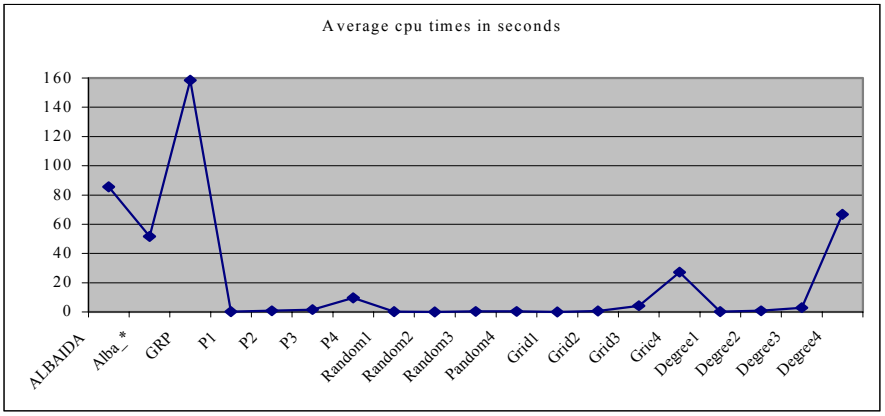

Figure 9 - Average CPU times per group

Finally, Figure 10, gives the percent contribution of each of the two phases to the overall time.

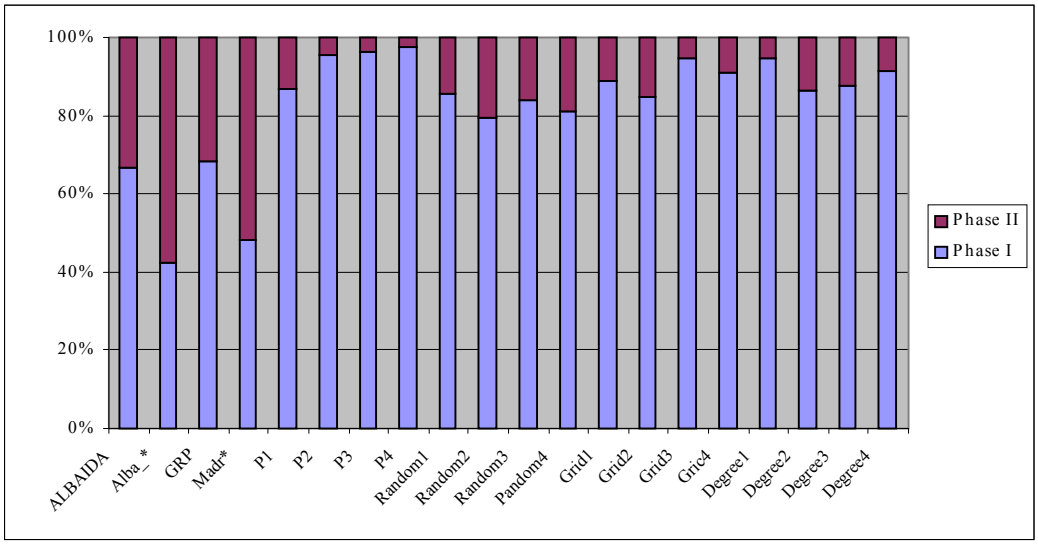

Figure 10. Percent contribution of each phase to overall time

7 Conclusions and Future Research

Our goal has been to explore the potential of the new formulations GW and FMGO for algorithm development for RPP. These formulations are fundamentally different those of the literature, all of which are broadly based on the early formulation CS. A key difference is that GW and FMGO make strong use of dominance relations. The algorithm we have proposed is quite rudimentary, as is appropriate for the first testing of a formulation. Even so, the computational results of Section 6 show that the algorithm is capable of solving almost all tested instances to proven optimality. Those not solved to optimality exhibit very small gaps between lower and upper bounds. Other positive aspects are the relatively small number of added inequalities needed to get good upper bounds, and the fact that the vast majority of these have efficient, exact separation procedures.

These results are especially encouraging in that there is a great deal of room for extensions. We hope and expect that, similar to what has already occurred with CS-based algorithms, future research will include investigation of adding efficiencies to algorithms based on the formulations GW and FMGO. One obvious possibility would be the addition of heuristic separation algorithms for Matching and Connectivity inequalities in Phase I to decrease the running time. A second involves further investigation of the Lagrangean relaxation of FMGO given in the appendix. This seems a natural way to solve FMGO, although the bounds have so far not led to advantages over the linear relaxation. Further research will entail heuristic methods for obtaining better Lagrange multipliers. A third area of investigation will be to search for additional valid inequalities for formulations GW and FMGO.

Acknowledgments

This research was partially funded by the Spanish Agency of Iberoamerican Cooperation (A.E.C.I.). Part of this work was carried out while the second author was on sabbatical leave at the Technical University of Catalonia, Spain. This support is gratefully acknowledged. The authors also want to thank Angel Corberán and Alain Hertz for providing us their

problem sets.

References

Christofides, N., 1976. Worst-Case Analysis of a New Heuristic for the Traveling Salesman Problem. Report No. 388, Graduate School of Industrial Administration, Carnegie-Mellon University.

Christofides, N., V. Campos, A. Corberán, E. Mota, 1981. An Algorithm for the Rural Postman Problem. Imperial College Report IC.O.R.81.5.

Corberán, A., A. Letchford, J. M. Sanchis, 1998. A Cutting Plane Algorithm for the General Routing Problem. Technical Report. Department of Statistics and Operations Research. University of Valencia, Spain.

Corberán, A., J. M. Sanchis, 1994. A Polyhedral Approach to the Rural Postman Problem. *European Journal of Operations Research* 79, 95-114.

Dantzig, G. B., D. R. Fulkerson, S. M. Johnson, 1954. Solution of a Large Scale Travelling Salesman Problem. *Operations Research 2*, 393-410.

Edmonds, J., 1965. Maximum Matching and a Polyhedron with 0,1-Vertices. *Journal of Research of the National Bureau of Standards* 69B, 125-130.

Edmonds, J., E. L. Johnson, 1973. Matchings, Euler Tours and the Chinese Postman. *Mathematical Programming* 5, 88-124.

Eiselt, H. A., M. Gendreau, G. Laporte, 1995a. Arc Routing Problems, Part I: The Chinese Postman Problem. *Operations Research* 43, 231-242.

Eiselt, H. A., M. Gendreau, G. Laporte, 1995b. Arc Routing Problems, Part II: The Rural Postman Problem. *Operations Research* 43, 399-414.

Finke, G., A. Claus, E. A.Gunn, 1984. A Two-Commodity Network Flow Approach to the Travelling Salesman Problem. *Congressus Numerantium* 41, 167-178.

Fischetti, M., G. Laporte, S. Martello, 1993. The Delivery Man Problem and Cumulative Matroids. *Operations Research* 41, 1055-1064.

Frederickson, G. N., 1979. Approximation Algorithms for Some Postman Problems.

Journal of the Association for Computing Machinery 7, 178-193.

Garfinkel, R., I. Webb, 1999. On Crossings, the Crossing Postman Problem, and the Rural Postman Problem. *Networks* 34, 173-180.

Ghiani, G., G. Laporte, 1999. A Branch-and-Cut Algorithm for the Undirected Rural Postman Problem. Center for research on Transportation. University of Montreal. Report G-97-65.

Gomory, R. E., T. C. Hu, 1961. Multi-Terminal Network Flows. *SIAM Journal* 9, 551-570.

Grötschel, M., O. Holland, 1985. Solving Matching Problems with Linear Programming. *Mathematical Programming* 33, 243-259.

Gusfield, D, 1990. Very Simple Methods for All Pairs Network Flow Analysis. *SIAM Journal on Computing* 19, 143-155.

Hertz, A., G. Laporte, P. Nanchen-Hugo, 1999. Improvement Procedures for the Undirected Rural Postman Problem. *INFORMS Journal on Computing* 1, 53-62.

Lenstra, J. K., A. H. G. Rinnooy Kan, 1976. On General Routing Problems. *Networks* 6, 273-280.

Letchford, A. N., 1997. New Inequalities for the General Routing Problem. *European Journal of Operations Research* 96, 317-332.

Lin, S., 1965. Computer Solutions of the Traveling Salesman Problem. *Bell System Technical Journal* 44, 2245-2269.

Padberg, M. W., M. R. Rao, 1982. Odd Minimum Cut-Sets and b-Matchings. *Mathematics of Operations Research* 7, 67-80.

Appendix - Lagrangean Relaxation for FMGO

Due to the structure of the problem that arises using GW or FMGO, the Lagrangean Relaxation technique seems a very good alternative for finding lower bounds or approximate solutions. In this case, by relaxing the "link" constraints we obtain two different and separate

problems, one on the set of x-variables (a perfect matching problem) and another on the set of y-variables (a network flow or a spanning tree problem). The problem of finding Lagrange multipliers, $\max_{\mu \geq 0} L(\mu)$, can be solved using either a classical technique such as the subgradient method, or by designing a heuristic that exploits the structure of the problem. Classical subgradient optimization has been used to solve the multiplier problem in the Lagrangean Relaxation. The bounds obtained so far are not better than those of Section 6. Moreover, the values for the Lagrangean relaxation are generally quite close to those of the linear relaxation. This leads us to believe that the existing gap between the Lagrangean relaxation and the optimal solution may be quite big. However, we know that in general, the values of the linear and Lagrangean relaxations are not the same and some heuristics should be investigated to obtain good Lagrangean multipliers.

Define variables $\mu_{k\ell} \geq 0$, associated with each constraint in (12a) of FMGO. The Lagrangean relaxation is:

$$\text{minimize} \sum_{\{i,j\} \in E_A} d_{ij} x_{ij} + \sum_{\substack{k=1,...,m \\ \ell=1,...,m \\ k \neq \ell}} \mu_{k\ell} (y_{k\ell} - (m-1) \sum_{\substack{(i,j) \in E_A^* \\ (i \in C_j \cap j \in C_k) \cup \\ (j \in C_\ell \cap i \in C_k)}} x_{ij})$$

s.t.

$$\begin{aligned}
\sum_{\{i,j\} \in E_k^+} x_{ij} &= 1, \quad k \in V_A \\
\sum_{\ell=2,...,m} y_{1\ell} &= m-1 \\
\sum_{k \neq t} y_{kt} - \sum_{k \neq t} y_{tk} &= 1, \quad t = 2, ..., m \\
x_{ij} &\in \{0,1\}, \quad \{i,j\} \in E_A \\
y_{k\ell} &\geq 0, \quad k \neq \ell,\ k = 1, ..., m,\ \ell = 2, ..., m
\end{aligned}$$