

**RELAXATIONS FOR MULTICOMMODITY CAPACITATED
NETWORK DESIGN PROBLEMS**

Bernard Gendron

Centre de recherche sur les transports
and Département d'informatique
et de recherche opérationnelle
Université de Montréal

Teodor Gabriel Crainic

Centre de recherche sur les transports
Université de Montréal
and Département des sciences administratives
Université du Québec à Montréal

February 1994

Abstract

This paper analyses classical relaxation methods applied to several formulations of a fixed charge multicommodity capacitated network design problem. The methods are compared theoretically on the basis of the relative strength of the relaxations. In order to assess more precisely the quality of the bounds generated by the various relaxations, feasible solutions are obtained via a heuristic based on the resource-decomposition principle. Computational results are presented and analyzed for a large set of test problems.

Key words : Multicommodity capacitated network design, relaxation methods.

Résumé

Dans cet article, nous analysons des méthodes classiques de relaxation appliquées à plusieurs formulations d'un problème de conception de réseau multiproduit avec capacités. Les méthodes sont comparées sur une base théorique en étudiant la force relative des relaxations. Afin de mesurer plus précisément la qualité des bornes générées par les diverses relaxations, des solutions réalisables sont obtenues par une heuristique basée sur le principe de la décomposition selon les ressources. Des résultats expérimentaux sont présentés et analysés pour un ensemble important de jeux de données.

Mots-clés : Conception de réseau multiproduit avec capacités, méthodes de relaxation.

1 Introduction

Network design models have been known for long as useful planning tools in areas such as transportation, telecommunications, manufacturing, logistics, among others. Comprehensive surveys on the applications of network design models and their resolution by mathematical programming techniques can be found in papers by Magnanti and Wong [19], Minoux [21], and Magnanti [16]. As pointed out by the last author, only particular versions of a general *fixed charge network design model* have been successively dealt with by using specialized methods for solving their continuous relaxations (Section 2 of the present paper contains a brief survey of the relevant literature). In particular, *multicommodity capacitated* versions of this general model pose considerable algorithmic challenges to researchers. The present paper can be seen as a first step towards the efficient resolution of large-scale multicommodity capacitated fixed charge network design problems. Its main objective is to propose and study bounding procedures based on classical relaxation approaches. Several methods are compared, both theoretically on the basis of the relative strength of the relaxations, and empirically by performing computational experiments on a large set of test problems. In order to assess more precisely the quality of the bounds generated by the various relaxations, feasible solutions of good quality are obtained via a heuristic based on the *resource-decomposition* principle, used in solving multicommodity network flow problems.

The paper is organized as follows. In Section 2, three formulations of the problem are presented. It is with respect to these formulations that various relaxations are defined and analyzed theoretically in Section 3. Section 4 is dedicated to algorithmic issues. We show, in particular, how to compute efficiently many of the bounds derived from the relaxations, and also present the resource-decomposition heuristic. Section 5 analyzes computational results. We summarize our results and discuss possible extensions to this work in Section 6.

2 Formulations

In this section, we present three formulations of a multicommodity capacitated fixed charge network design problem. The first formulation, called *weak formulation*, is sufficient to describe the problem completely, but, as evidenced from past studies on related difficult mixed-integer problems, it may not be a good modeling choice for computing tight bounds through relaxation approaches. Better alternatives are provided by the *strong formulation* and the *extended formulation*, that are presented next. By adding valid inequalities to any of these formulations, it is possible to further strengthen them. We present a class of such valid inequalities, called *knapsack inequalities*. To conclude this section, we give a brief survey of the literature on related problems and formulations.

2.1 Weak Formulation

We are given a *directed network* $G = (N, A)$, where N is the set of nodes and A is the set of arcs, on which several commodities, represented by set P , are moving. For each commodity p , the set of nodes may be partitioned into three subsets: $O(p)$, the set of nodes which send more flow of commodity p than the quantity they receive, called *origin nodes*; $D(p)$, the set of nodes which receive more flow of commodity p than the quantity they send, called *destination nodes*; $T(p)$, the set of nodes which receive and send the same quantity of flow of commodity p , called *transshipment nodes*. For each commodity p , we associate to each origin $i \in O(p)$ a positive *supply* o_i^p and to each destination $i \in D(p)$ a positive *demand* d_i^p . We assume that $\sum_{i \in O(p)} o_i^p = \sum_{i \in D(p)} d_i^p = t_p$ for each commodity p . To each arc (i, j) , we associate a positive *total capacity* u_{ij} on the amount of flow of all commodities that can move through the arc, and nonnegative *partial capacities* b_{ij}^p on the amount of flow of commodity p that can move through the arc. Without loss of generality, we assume that $b_{ij}^p \leq \min(u_{ij}, t_p)$, for each arc (i, j) and each commodity p , and that $u_{ij} \leq \sum_{p \in P} b_{ij}^p$, for each arc (i, j) . A nonnegative *routing cost* c_{ij}^p is incurred for each unit of flow of commodity p that moves through arc (i, j) . Moreover, a nonnegative *fixed cost* f_{ij} is added when any amount of flow moves through arc (i, j) . The problem consists in minimizing the sum of all costs, while satisfying supply and demand requirements, flow

conservation at transshipment nodes, and capacity constraints.

To formulate the problem, we define continuous nonnegative *flow variables* x_{ij}^p , which represent the amount of flow of commodity p moving on arc (i, j) , and binary *design variables* y_{ij} , defined as follows:

$$y_{ij} = \begin{cases} 0, & \text{if } x_{ij}^p = 0, \forall p \in P \\ 1, & \text{otherwise} \end{cases} \quad \forall (i, j) \in A \quad (1)$$

We also introduce the following notation to describe the sets of outward and inward neighbors of any node i : $N^+(i) = \{j \in N \mid (i, j) \in A\}$; $N^-(i) = \{j \in N \mid (j, i) \in A\}$.

The *weak formulation*, noted W , may then be written as follows:

$$Z(W) = \min \sum_{(i,j) \in A} \sum_{p \in P} c_{ij}^p x_{ij}^p + \sum_{(i,j) \in A} f_{ij} y_{ij} \quad (2)$$

- *Flow conservation constraints*

$$\sum_{j \in N^+(i)} x_{ij}^p - \sum_{j \in N^-(i)} x_{ji}^p = \begin{cases} o_i^p & \text{if } i \in O(p) \\ -d_i^p & \text{if } i \in D(p) \\ 0 & \text{if } i \in T(p) \end{cases} \quad \forall i \in N, p \in P \quad (3)$$

- *Total capacity constraints*

$$\sum_{p \in P} x_{ij}^p \leq u_{ij} \quad \forall (i, j) \in A \quad (4)$$

- *Partial capacity constraints*

$$x_{ij}^p \leq b_{ij}^p \quad \forall (i, j) \in A, p \in P \quad (5)$$

- *Nonnegativity constraints on flow variables*

$$x_{ij}^p \geq 0 \quad \forall (i, j) \in A, p \in P \quad (6)$$

- *Weak linking constraints*

$$\sum_{p \in P} x_{ij}^p \leq u_{ij} y_{ij} \quad \forall (i, j) \in A \quad (7)$$

- *Bounding constraints*

$$y_{ij} \leq 1 \quad \forall (i, j) \in A \quad (8)$$

- *Nonnegativity constraints on design variables*

$$y_{ij} \geq 0 \quad \forall (i, j) \in A \quad (9)$$

- *Integrality constraints*

$$y_{ij} \text{ integer} \quad \forall (i, j) \in A \quad (10)$$

Note that, for the sake of clarity, we have chosen to include the total capacity constraints 4 into this formulation, although they are redundant, being implied by the weak linking constraints 7 and the bounding constraints 8.

Note also that uncapacitated network design problems can be modeled with this formulation: set, for example, $b_{ij}^p = t_p \quad \forall (i, j) \in A, p \in P$ and $u_{ij} = \sum_{p \in P} b_{ij}^p \quad \forall (i, j) \in A$. These bounds can be made tighter in some cases, such as the one of bipartite networks, where all origins and destinations are the same for every commodity, and there are no transshipment nodes. Then, we may set $b_{ij}^p = \min(o_i^p, d_j^p) \quad \forall (i, j) \in A, p \in P$ and $u_{ij} = \sum_{p \in P} b_{ij}^p \quad \forall (i, j) \in A$.

In this paper, we focus only on capacitated problems, that is, instances for which there exists at least one arc (i, j) such that $u_{ij} < \sum_{p \in P} b_{ij}^p$. Even in these cases, partial capacities may not be true capacities, but rather represent valid upper bounds on the amount of flow of given commodities that can move through given arcs.

2.2 Strong Formulation

The *strong formulation*, noted S (with optimal value $Z(S)$), is obtained from the weak formulation by adding the following constraints:

- *Strong linking constraints*

$$x_{ij}^p \leq b_{ij}^p y_{ij} \quad \forall (i, j) \in A, p \in P \quad (11)$$

These constraints being implied by relations 5 to 10, they are valid inequalities for the weak formulation, and thus we have $Z(W) = Z(S)$.

2.3 Extended Formulation

Following the work of Rardin and Choe [26], and Rardin [25], we reformulate the problem by decomposing the flow on each arc by origin-destination pairs. More precisely, we define the continuous nonnegative flow variables w_{ij}^{pkl} , which represent the amount of flow of commodity p moving on arc (i, j) , originating from $k \in O(p)$ and destined for $l \in D(p)$. The *extended formulation*, noted E , may then be written as follows:

$$Z(E) = \min \sum_{(i,j) \in A} \sum_{p \in P} \sum_{k \in O(p)} \sum_{l \in D(p)} c_{ij}^p w_{ij}^{pkl} + \sum_{(i,j) \in A} f_{ij} y_{ij} \quad (12)$$

- *Flow conservation constraints*

$$\sum_{l \in D(p)} \sum_{j \in N^+(k)} w_{kj}^{pkl} = o_k^p \quad \forall p \in P, k \in O(p) \quad (13)$$

$$\sum_{k \in O(p)} \sum_{j \in N^-(l)} w_{jl}^{pkl} = d_l^p \quad \forall p \in P, l \in D(p) \quad (14)$$

$$\sum_{j \in N^+(i)} w_{ij}^{pkl} - \sum_{j \in N^-(i)} w_{ji}^{pkl} = 0 \quad \forall p \in P, k \in O(p), l \in D(p), i \in N - \{k, l\} \quad (15)$$

- *Total capacity constraints*

$$\sum_{p \in P} \sum_{k \in O(p)} \sum_{l \in D(p)} w_{ij}^{pkl} \leq u_{ij} \quad \forall (i, j) \in A \quad (16)$$

- *Partial capacity constraints*

$$\sum_{k \in O(p)} \sum_{l \in D(p)} w_{ij}^{pkl} \leq b_{ij}^p \quad \forall (i, j) \in A, p \in P \quad (17)$$

- *Nonnegativity constraints on flow variables*

$$w_{ij}^{pkl} \geq 0 \quad \forall (i, j) \in A, p \in P, k \in O(p), l \in D(p) \quad (18)$$

- *Weak linking constraints*

$$\sum_{p \in P} \sum_{k \in O(p)} \sum_{l \in D(p)} w_{ij}^{pkl} \leq u_{ij} y_{ij} \quad \forall (i, j) \in A \quad (19)$$

- *Strong linking constraints*

$$\sum_{k \in O(p)} \sum_{l \in D(p)} w_{ij}^{pkl} \leq b_{ij}^p y_{ij} \quad \forall (i, j) \in A, p \in P \quad (20)$$

- *Extended linking constraints*

$$\sum_{l \in D(p)} w_{ij}^{pkl} \leq \min(b_{ij}^p, o_k^p) y_{ij} \quad \forall (i, j) \in A, p \in P, k \in O(p) \quad (21)$$

$$\sum_{k \in O(p)} w_{ij}^{pkl} \leq \min(b_{ij}^p, d_l^p) y_{ij} \quad \forall (i, j) \in A, p \in P, l \in D(p) \quad (22)$$

- *Bounding constraints* (8)
- *Nonnegativity constraints on design variables* (9)
- *Integrality constraints* (10)

Proposition 1

$$Z(W) = Z(S) = Z(E)$$

Proof:

We only prove the last equality, since the first one was established in Section 2.2. We show that to any optimal solution to E corresponds a feasible solution to S with the same cost, and vice-versa.

We first note that the extended linking constraints can be removed from formulation E since they are implied by other constraints. Now, given an optimal solution (w, y) to E , we note that no flow may follow a cycle, since all costs are nonnegative. Hence, in particular, $\sum_{j \in N^-(k)} w_{jk}^{pkl} = 0$ and $\sum_{j \in N^+(l)} w_{lj}^{pkl} = 0 \quad \forall p \in P, k \in O(p), l \in D(p)$. Then, one may easily show that the transformation $x_{ij}^p = \sum_{k \in O(p)} \sum_{l \in D(p)} w_{ij}^{pkl} \quad \forall (i, j) \in A, p \in P$ identifies a feasible solution (x, y) to S with cost $Z(E)$.

Conversely, given an optimal solution (x, y) to S , we can obtain flow values on the paths which connect, for each commodity, every origin to every destination. This follows from the equivalence between the arc and path formulations for network flow problems

(see, for example, Ahuja, Magnanti and Orlin [1]), and the fact that all costs are nonnegative. Moreover, these path flows have a unique representation in terms of the variables w . The solution (w, y) thus obtained clearly satisfies every constraint of formulation E and has a cost equal to $Z(S)$. \square

2.4 Knapsack Inequalities

Several valid inequalities can be added to any of the previous formulations in order to strengthen them. Padberg, Van Roy and Wolsey [24], for example, have derived a class of valid inequalities, called flow cover inequalities, for a single-node flow problem which can be seen as a relaxation for any of the three formulations. These inequalities involve both flow and design variables, and have been used by Van Roy and Wolsey [30] in a linear programming-based cutting plane approach. Here, we take a different point of view. Our aim is to derive relaxation methods that exploit the network structure of the problem without necessarily dropping the integrality constraints on the design variables. To that aim, we allow ourselves to add to the formulations a class of valid inequalities (or any valid inequalities derived from them), called *knapsack inequalities*. These inequalities have also been studied by Padberg, Van Roy and Wolsey, and were shown to be helpful in defining strong relaxations for the capacitated plant location problem, where they reduce to a single inequality (see Nauss [23], and Cornuejols, Sridharan and Thizy [2]).

Given the following notations: $O = \cup_{p \in P} O(p)$, $D = \cup_{p \in P} D(p)$ and $P^O(i) = \{p \in P \mid i \in O(p)\}$, $P^D(j) = \{p \in P \mid j \in D(p)\}$, the knapsack inequalities may be stated as follows:

- *Weak knapsack constraints*

$$\sum_{j \in N^+(i)} u_{ij} y_{ij} \geq \sum_{p \in P^O(i)} \alpha_i^p \quad \forall i \in O \quad (23)$$

$$\sum_{i \in N^-(j)} u_{ij} y_{ij} \geq \sum_{p \in P^D(j)} d_j^p \quad \forall j \in D \quad (24)$$

- *Strong knapsack constraints*

$$\sum_{j \in N^+(i)} b_{ij}^p y_{ij} \geq \alpha_i^p \quad \forall i \in O(p), p \in P \quad (25)$$

$$\sum_{i \in N^-(j)} b_{ij}^p y_{ij} \geq d_j^p \quad \forall j \in D(p), p \in P \quad (26)$$

Weak and strong knapsack constraints are derived, respectively, from weak and strong linking constraints, plus flow conservation constraints and nonnegativity restrictions on flow variables.

2.5 Related Problems and Formulations

Other multicommodity capacitated network design models may be found in the literature, but few methods have been presented to solve them.

Rardin and Choe [26], and Rardin [25] propose a model similar to the weak formulation, but with no partial capacities. In order to obtain tighter relaxations, they then introduce two reformulations of the problem, one based on a path representation rather than a node-arc structure, and another similar to the extended formulation. They then focus only on uncapacitated problems and show that, in this case, the continuous relaxation of the extended formulation dominates the continuous relaxation of the path formulation. They also give numerical results relative to a Lagrangean relaxation approach with respect to the linking constraints. Helgason [10], in an unpublished Ph.D. dissertation, studies a model which is very close to the strong formulation and proposes a Lagrangean relaxation approach with respect to the flow conservation constraints. He reports disappointing results on a real problem arising in airplane logistics.

Another multicommodity capacitated network design problem that has received attention recently is the network loading problem. It is defined on an undirected network and has a structure similar to the weak formulation, but with the following important distinctions: there are no routing costs; there is a single origin and a single destination per commodity; there are no bounding constraints on the design variables. In this problem, design variables do not only represent decisions on the inclusion of arcs in the network design, but also a number of facilities to install on each arc. Magnanti, Mirchandani and Vachani [17, 18] study the polyhedral structure of this problem and propose a Lagrangean relaxation with respect to the flow conservation constraints.

3 Relaxations

With respect to computational effort involved in the resolution of any of the three formulations presented in the previous section, two points of view may be considered. First, that the presence of the integrality constraints, without which all three formulations may be solved as linear programming problems, is the major difficulty. The other point of view considers that the complications arise from the interactions between the flow conservation and the linking constraints. To address these issues, various relaxation approaches may be used: continuous relaxations, Lagrangean relaxations, Lagrangean decompositions.

In this section, we define and compare theoretically several problems obtained by applying these relaxation methods. We present four types of results. First, we measure the strength of the three formulations with respect to the various types of relaxations. Then, we order the relaxations for each formulation. The third category of results considers the addition of valid inequalities derived from the knapsack constraints to each relaxation. Finally, we examine the effect of removing capacity constraints from the relaxations. Most of these results are obtained by using the primal interpretation of Lagrangean relaxation due to Geoffrion [6], and its specialization to Lagrangean decomposition due to Guignard and Kim [9].

3.1 Definitions

For each of the three formulations, we obtain *continuous relaxations* by removing the integrality constraints. The resulting problems are noted CW , CS and CE , for the continuous relaxations of, respectively, the weak, the strong and the extended formulations.

We define two types of *Lagrangean relaxations*: one by relaxing the linking constraints, the other by relaxing the flow conservation constraints. When the linking constraints are relaxed in any of the formulations, the resulting Lagrangean subproblems present, for a fixed value of the multiplier vector, a network flow structure. Hence, these subproblems are called *flow subproblems*, and the corresponding Lagrangean duals are identified, respectively, by FW , FS and FE . Similarly, when the flow conservation constraints are relaxed in each of the three formulations, we call the resulting Lagrangean formulations

linking subproblems and identify the associated Lagrangean duals as LW , LS and LE , respectively.

In order to apply *Lagrangean decomposition*, we first duplicate every variable in each respective formulation. We then define two subproblems as follows. In terms of the original variables, the first subproblem retains every constraint, except the linking ones. In terms of the new variables, we duplicate every constraint, except the flow conservation constraints, and, as a result, derive the second subproblem. The resulting structures are identical to those obtained by using the defined Lagrangean relaxations: the first subproblem is a flow subproblem, while the second one is a linking subproblem. The Lagrangean duals associated with the Lagrangean decompositions relative to each of the three formulations are noted DW , DS and DE , respectively.

3.2 Relative Strength of the Relaxations

In this section, we use the following notation. The optimal value of a problem X is noted $Z(X)$. The set of points satisfying constraints k_1, k_2, \dots, k_n is noted $(k_1 - k_2 - \dots - k_n)$, while its convex hull is noted $Co(k_1 - k_2 - \dots - k_n)$.

The first result shows that, as expected, the extended formulation gives the tightest relaxations, while the strong formulation is better than the weak one only for continuous relaxations and Lagrangean relaxations with respect to the linking constraints.

Proposition 2

$$a) Z(CW) \leq Z(CS) \leq Z(CE)$$

$$b) Z(FW) \leq Z(FS) \leq Z(FE)$$

$$c) Z(LW) = Z(LS) \leq Z(LE)$$

$$d) Z(DW) = Z(DS) \leq Z(DE)$$

Proof:

The first inequality in part a) follows from the observation that CW is derived from CS by removing the strong linking constraints. To show the second inequality, we define a relaxation of CE by removing the extended linking constraints. By the same arguments

as in the proof of Proposition 1, we can show that this relaxation has an optimal value equal to $Z(CS)$. Thus, the inequality follows.

To prove part b), we first note that $Z(FS)$ can be obtained by minimizing the objective function of formulation S over the set $(7 - 11) \cap Co(3 - 4 - 5 - 6 - 8 - 9 - 10)$. The relaxation of this problem defined by removing constraints 11 has an optimal value equal to $Z(FW)$, and the first inequality follows. To prove the second inequality, note that $Z(FE)$ can be derived by minimizing the objective function of formulation E over the set $(19 - 20 - 21 - 22) \cap Co(13 - 14 - 15 - 16 - 17 - 18 - 8 - 9 - 10)$. The second component of this set separates into two subsets: one that depends on the continuous flow variables and another that depends on the design variables. Therefore, $Z(FE)$ can equivalently be obtained by minimizing the objective function of E over the set $(19 - 20 - 21 - 22 - 13 - 14 - 15 - 16 - 17 - 18) \cap Co(8 - 9 - 10)$. Consider a relaxation of this problem defined by removing constraints 21 and 22. Again, by using the same arguments as in the proof of Proposition 1, we can show that the optimal value of this relaxation can be derived by minimizing the objective function of formulation S over the set $(7 - 11 - 3 - 4 - 5 - 6) \cap Co(8 - 9 - 10)$, and is equal to $Z(FS)$.

To prove the equality in part c), note that $Z(LS)$ can be derived by minimizing the objective function of S over the set $(3) \cap Co(4 - 5 - 6 - 7 - 11 - 8 - 9 - 10)$. Since constraints 11 are implied by constraints 5 to 10, this set is equivalent to $(3) \cap Co(4 - 5 - 6 - 7 - 8 - 9 - 10)$, and minimizing the objective function of W over this set gives $Z(LW)$. To show the inequality in part c), we use the relation $Z(FS) \leq Z(FE)$ of part b), and the equalities $Z(FS) = Z(LS)$ and $Z(FE) = Z(LE)$, which are stated and proved in Proposition 3.

Finally, to prove part d), we use part c) and the relations $Z(LS) = Z(DS)$ and $Z(LE) = Z(DE)$, which are also stated and proved in Proposition 3. \square

An easier way to prove part b) is to notice that the flow subproblems in every formulation possess the integrality property of Geoffrion [6], and the associated Lagrangean duals provide the same bound as the continuous relaxations. However, the argument in the proof is to be preferred since it can be easily generalized to the case where valid

inequalities derived from the knapsack constraints are added to the formulations.

We now give one of the important results of this section, which states that, with respect to the strong and the extended formulations, no improvement of the bounds computed by the continuous relaxations can be obtained by using any of the Lagrangean relaxations or decompositions that we have defined previously.

Proposition 3

a) $Z(CW) = Z(FW) \leq Z(LW) = Z(DW)$

b) $Z(CS) = Z(FS) = Z(LS) = Z(DS)$

c) $Z(CE) = Z(FE) = Z(LE) = Z(DE)$

Proof:

We first prove part b). The first equality is established by noting, as in the proof of Proposition 2, that $Z(FS)$ can be derived by minimizing the objective function of S over the set $(7 - 11 - 3 - 4 - 5 - 6) \cap Co(8 - 9 - 10)$. Since $Co(8 - 9 - 10) = (8 - 9)$, the equality follows.

We now prove the second equality. Using the arguments in the proof of Proposition 2, we only have to establish the following equality: $(4 - 5 - 6 - 7 - 11) \cap Co(8 - 9 - 10) = Co(4 - 5 - 6 - 7 - 11 - 8 - 9 - 10)$. The inclusion \supseteq is trivial. To prove \subseteq , note that $Q = Co(4 - 5 - 6 - 7 - 11 - 8 - 9 - 10)$ is the largest polyhedron contained in $T = (4 - 5 - 6 - 7 - 11 - 8 - 9)$ for which every extreme point satisfies $y_{ij} \in \{0, 1\}$, $\forall (i, j) \in A$. Therefore, it is sufficient to show that every extreme point of $R = (4 - 5 - 6 - 7 - 11) \cap Co(8 - 9 - 10)$ satisfies $y_{ij} \in \{0, 1\}$, $\forall (i, j) \in A$, since $R \subseteq T$ (in fact, $R = T$). Suppose the contrary: there exists an extreme point $(x(0), y(0))$ of R such that $y(0)$ has at least one fractional component. As a consequence, $y(0)$ cannot be an extreme point of H , the projection of $Co(8 - 9 - 10)$ onto the y -space. Thus, there exist $y(1), y(2) \in H$ such that $y(0) = \frac{1}{2}y(1) + \frac{1}{2}y(2)$ and $y(1) \neq y(2)$. Define $x(k)$, $k = 1, 2$, as follows:

$$x_{ij}^p(k) = \begin{cases} 0 & \text{if } y_{ij}(0) = 0 \\ x_{ij}^p(0) \left(\frac{y_{ij}(k)}{y_{ij}(0)} \right) & \text{if } y_{ij}(0) > 0 \end{cases} \quad \forall (i, j) \in A, p \in P$$

Then, $(x(k), y(k)) \in R$, and, by construction, $x(0) = \frac{1}{2}x(1) + \frac{1}{2}x(2)$. Hence, $(x(0), y(0))$ is not an extreme point of R , which contradicts the hypothesis. As a consequence, we have $Z(FS) = Z(LS)$.

We now show the last equality in part b). We first note that $Z(DS)$ can be obtained by minimizing the objective function of S over the set $Co(3 - 4 - 5 - 6 - 8 - 9 - 10) \cap Co(4 - 5 - 6 - 7 - 11 - 8 - 9 - 10)$, which is equivalent, by arguments previously used, to $(3 - 4 - 5 - 6) \cap Co(8 - 9 - 10) \cap Co(4 - 5 - 6 - 7 - 11 - 8 - 9 - 10)$. Thus, using the primal interpretation of $Z(LS)$ and the notation $U = Co(4 - 5 - 6 - 7 - 11 - 8 - 9 - 10)$, it is sufficient to show that $U = (4 - 5 - 6) \cap Co(8 - 9 - 10) \cap U$. The inclusion \supseteq is trivial, while the inclusion \subseteq follows from the fact that $U \subseteq (4 - 5 - 6) \cap Co(8 - 9 - 10)$. This shows that $Z(LS) = Z(DS)$.

To prove part a), we note that the first equality can be established in a similar way as the one in part b), while the other relations are direct consequences of part b) and Proposition 2. The proof of part c) is similar to that of part b) and is, therefore, omitted. \square

In spite of these results, it is possible to improve over the bounds given by the continuous relaxations, by adding valid inequalities derived from the knapsack constraints. Let $K = (23 - 24 - 25 - 26)$, and let X_{K^*} denote the problem obtained from problem X by adding the following constraints:

$$y \in K^*, \quad K^* \supseteq K \tag{27}$$

In the case of a formulation obtained by using Lagrangean decomposition, these constraints may be added either to the flow subproblem, or to the linking subproblem, or to both (with the appropriate change of variables). We denote the Lagrangean duals corresponding to each of these three options by, respectively, $X_{K^*}(F)$, $X_{K^*}(L)$ and $X_{K^*}(F, L)$, where X stands for DW , DS or DE . Our first result with respect to adding knapsack inequalities states that the three approaches provide the same bound:

Proposition 4

a) $Z(DW_{K^*}(F)) = Z(DW_{K^*}(L)) = Z(DW_{K^*}(F, L))$

$$b) Z(DS_{K^*}(F)) = Z(DS_{K^*}(L)) = Z(DS_{K^*}(F, L))$$

$$c) Z(DE_{K^*}(F)) = Z(DE_{K^*}(L)) = Z(DE_{K^*}(F, L))$$

Proof:

We prove only part b), since parts a) and c) are derived similarly. Using the obvious inequalities $Z(LS_{K^*}) \leq Z(DS_{K^*}(F)) \leq Z(DS_{K^*}(F, L))$ and $Z(LS_{K^*}) \leq Z(DS_{K^*}(L)) \leq Z(DS_{K^*}(F, L))$, it is sufficient to show that $Z(LS_{K^*}) = Z(DS_{K^*}(F, L))$. We omit the details of the proof, since the arguments are similar to the ones used to show $Z(LS) = Z(DS)$ in the proof of Proposition 3. \square

Since, by Proposition 4, the three Lagrangean duals provide the same bound, we use a unique notation $Z(X_{K^*})$, for the optimal value of the three problems $X_{K^*}(F)$, $X_{K^*}(L)$ and $X_{K^*}(F, L)$, where X stands for one of the symbols DW , DS or DE .

The next proposition states that the addition of valid inequalities derived from the knapsack constraints can improve over the bounds obtained by the defined Lagrangean methods. This is not true however for the continuous relaxations of the strong and the extended formulations, since in these cases, the knapsack constraints are redundant. For the continuous relaxation of the weak formulation, improvement is still possible by adding inequalities derived from the strong knapsack constraints.

Proposition 5

$$a) Z(CW) \leq Z(CW_{K^*}), Z(CS) = Z(CS_{K^*}), Z(CE) = Z(CE_{K^*})$$

$$b) Z(FW) \leq Z(FW_{K^*}), Z(FS) \leq Z(FS_{K^*}), Z(FE) \leq Z(FE_{K^*})$$

$$c) Z(LW) \leq Z(LW_{K^*}), Z(LS) \leq Z(LS_{K^*}), Z(LE) \leq Z(LE_{K^*})$$

$$d) Z(DW) \leq Z(DW_{K^*}), Z(DS) \leq Z(DS_{K^*}), Z(DE) \leq Z(DE_{K^*})$$

The two following propositions generalize Propositions 2 and 3. The proof of Proposition 6 follows exactly that of Proposition 2 and is, therefore, omitted. Proposition 7 can be derived by similar arguments as in the proof of Proposition 3, by noting, in addition, that the flow subproblems with the additional constraints 27 do not necessarily satisfy the integrality property.

Proposition 6

- a) $Z(CW_{K^*}) \leq Z(CS_{K^*}) \leq Z(CE_{K^*})$
- b) $Z(FW_{K^*}) \leq Z(FS_{K^*}) \leq Z(FE_{K^*})$
- c) $Z(LW_{K^*}) = Z(LS_{K^*}) \leq Z(LE_{K^*})$
- d) $Z(DW_{K^*}) = Z(DS_{K^*}) \leq Z(DE_{K^*})$

Proposition 7

- a) $Z(CW_{K^*}) \leq Z(FW_{K^*}) \leq Z(LW_{K^*}) = Z(DW_{K^*})$
- b) $Z(CS_{K^*}) \leq Z(FS_{K^*}) = Z(LS_{K^*}) = Z(DS_{K^*})$
- c) $Z(CE_{K^*}) \leq Z(FE_{K^*}) = Z(LE_{K^*}) = Z(DE_{K^*})$

Our last result states that the total capacity constraints can be removed from any of the defined relaxations without changing the bounds. This result is trivial for continuous relaxations and Lagrangean relaxations with respect to the flow conservation constraints since, in these cases, the total capacity constraints are redundant in the resulting subproblems. This observation is not true, however, for the flow subproblems that arise from Lagrangean decompositions and Lagrangean relaxations with respect to the linking constraints. Let X be a Lagrangean dual derived from one of these two types of relaxations. We note by X^T the problem obtained from X by removing the total capacity constraints from the flow subproblem, and by X^{TP} the problem obtained from X by removing both the total and the partial capacity constraints from the flow subproblem.

Proposition 8

- a) $Z(FW) = Z(FW^T), Z(DW) = Z(DW^T)$
- b) $Z(FS) = Z(FS^{TP}), Z(DS) = Z(DS^{TP})$
- c) $Z(FE) = Z(FE^{TP}), Z(DE) = Z(DE^{TP})$

Proof:

We only show that $Z(FS) = Z(FS^{TP})$, since all other relations can be derived in a similar way. We note that $Z(FS^{TP})$ can be obtained by minimizing the objective function of S over the set $(7 - 11) \cap Co(3 - 6 - 8 - 9 - 10)$, which is equivalent to $(7 - 11 - 3 - 6) \cap$

$Co(8 - 9 - 10)$. Therefore, using the primal interpretation of $Z(FS)$, it is sufficient to show that $(7 - 11 - 4 - 5) \cap Co(8 - 9 - 10) = (7 - 11) \cap Co(8 - 9 - 10)$. This relation follows from the observation that constraints 4 and 5 are implied by 7, 11 and 8, and the fact that $(7 - 11) \cap Co(8 - 9 - 10) = (7 - 11 - 8) \cap Co(8 - 9 - 10)$. \square

It is noteworthy that the argument used in this proof extends easily to the case where valid inequalities derived from the knapsack constraints are added to the formulations. Therefore, it is always true that removing the total capacity constraints from the relaxations do not change the resulting bounds.

4 Algorithmic Issues

This section is dedicated to an extensive discussion of algorithmic and computational issues related to the determination of bounds based on the relaxations defined in Section 3. We aim both to identify the most promising algorithmic approaches for each one of the problem types identified in the previous section, and to present our implementation used in Section 5.

The presentation is organized as follows. We first examine the computational structures derived from the continuous relaxations, and then study the two main classes of subproblems that arise from Lagrangean methods: the flow and the linking subproblems. The theoretical development of Section 3 has shown that all these subproblems may be strengthened by adding valid inequalities derived from the knapsack constraints. We identify such a class of valid inequalities, and show that the resulting subproblems decompose into 0-1 knapsack problems.

The presentation is then oriented towards the development of an efficient upper bound heuristic method, that makes use of the resource-decomposition principle. Finally, we discuss the implementation of bounding procedures, based on subgradient optimization, which combine Lagrangean methods to the upper bounding heuristic.

4.1 Continuous Relaxations

The continuous relaxations of the three formulations can be solved by any general linear programming method. This approach suffers, however, from two major drawbacks. First, the network structure of the problem is not exploited. Second, the potentially huge number of linking constraints, particularly for the extended formulation, does not permit an efficient resolution of large instances. We now look at each of the three continuous relaxations and try to identify solution approaches that avoid these drawbacks.

The continuous relaxation of the weak formulation is most efficiently solved by reformulating it as a multicommodity network flow problem. To derive this reformulation, we observe that, since all fixed costs are nonnegative, there is an optimal solution that satisfies every linking constraint at equality. Therefore, we can get rid of the design variables and obtain the following problem: $Z(CW) = \min \sum_{(i,j) \in A} \sum_{p \in P} (c_{ij}^p + f_{ij}/u_{ij}) x_{ij}^p$, subject to constraints 3, 4, 5 and 6. Still, the problem remains computationally difficult, as evidenced by past studies on multicommodity network flow problems (see the book by Ahuja, Magnanti and Orlin [1], for a recent survey of methods for this class of problems).

The continuous relaxation of the strong formulation is even more difficult to solve. One approach, that attempts to exploit the network structure, consists in first solving, by a specialized version of the simplex method, the single-commodity network flow problems obtained once the total capacity and the linking constraints are removed. This solution may then be used either as an initial basis to a dual simplex method (this approach is implemented in the commercial software CPLEX [3]), or as the starting point to a strategy that tries to add gradually only linking constraints that are violated by the current solution (such a strategy was tested by Morris [22] for the uncapacitated plant location problem). Another approach, that tries to reduce the impact of the large number of strong linking constraints, is to treat them implicitly as variable upper bound constraints (for adaptations of linear programming methods to handle variable upper bounds, see, for example, the works of Schrage [27], and Todd [28] on simplex methods, and those of Todd [29], and Hurd and Murphy [11] on interior-point approaches).

Adapting these ideas to the continuous relaxation of the extended formulation poses

considerable difficulties since, not only the number of linking constraints may be significantly higher than for the continuous relaxations of the two other formulations, but the number of variables as well. Yet, we can identify two ways of exploiting the network structure of the problem, in order to obtain a good initial basis for the dual simplex method. The first approach consists in removing the total capacity, the partial capacity, and the linking constraints. Then, as pointed out by Rardin and Choe [26], and Rardin [25], the resulting problem reduces to the determination of a shortest path for every origin-destination pair, followed by the resolution of $|P|$ transportation problems. A second approach consists in solving the single-commodity network flow problems obtained from the strong formulation after removing the total capacity and the linking constraints. Contrary to the solution obtained by the first approach, this one takes into account the partial capacity constraints. It is expressed, however, in the space of x variables. To translate it in the space of the w variables, we can adapt an algorithm to obtain a path representation of a solution from an arc representation (see Ahuja, Magnanti and Orlin [1], for such an algorithm).

4.2 Flow Subproblems

The flow subproblem arises from applying Lagrangean decomposition or relaxation with respect to the linking constraints to any of the three formulations. It decomposes into two parts: one that depends only on the flow variables, and another that depends only on the design variables. This second part of the subproblem is solvable by inspection, unless valid inequalities derived from the knapsack constraints are added to the formulation. This case is discussed in Section 4.4. As for the part of the subproblem that depends only on the flow variables, several algorithmic alternatives are possible, as suggested by Proposition 8. For each of the three formulations, we now study several of these alternatives.

For the weak formulation, one alternative is to retain the total capacity constraints. The resulting subproblem is a multicommodity network flow problem. If the subproblem is obtained by using Lagrangean relaxation, it is sufficient to solve only one multicommod-

ity network flow problem in order to find the optimal value of the Lagrangean dual. This follows from the fact that the Lagrangean dual has the same optimal value as the continuous relaxation, which can be solved as a multicommodity network flow problem. Moreover, if we denote by α_{ij} the Lagrangean multiplier associated to each weak linking constraint, an optimal solution to the Lagrangean dual is given by $\alpha_{ij} = f_{ij}/u_{ij} \forall (i, j) \in A$. Another alternative is to remove the total capacity constraints. The resulting subproblem decomposes into $|P|$ single-commodity network flow problems.

The same options can be used for the strong formulation, along with a third approach, which consists in removing the total and the partial capacity constraints. The resulting subproblem decomposes into $|P|$ single-commodity uncapacitated network flow problems. This alternative may be particularly advantageous if there is a single origin or a single destination per commodity, in which case the approach further decomposes into a series of shortest path problems, one for each origin-destination pair. If there are multiple origins and multiple destinations per commodity, $|P|$ transportation problems must be solved in addition to the shortest path problems.

For the extended formulation, it seems that the only approach that leads to a computationally tractable structure is to remove the total and the partial capacity constraints. There is a notable exception, however, when the flow subproblem is derived from an *aggregated Lagrangean decomposition*. This scheme is obtained by modifying slightly the definition of the Lagrangean decomposition given in Section 3.1: instead of duplicating each flow variable, copy constraints on aggregations of flow variables over all origins and all destinations are introduced. Let σ_{ij}^p be the associated Lagrangean multiplier. When the part of the flow subproblem that depends only of the flow variables is isolated, the following structure appears: $Z(AD) = \min \sum_{(i,j) \in A} \sum_{p \in P} \sum_{k \in O(p)} \sum_{l \in D(p)} (c_{ij}^p + \sigma_{ij}^p) w_{ij}^{pkl}$, subject to constraints 13, 14, 15, 17 and 18. One can then solve an equivalent problem in the space of x variables, which decomposes into $|P|$ single-commodity network flow problems: $Z(AD) = \min \sum_{(i,j) \in A} \sum_{p \in P} (c_{ij}^p + \sigma_{ij}^p) x_{ij}^p$, subject to constraints 3, 5 and 6. The optimal solution can then be translated in the space of the w variables. By using this scheme, the partial capacity constraints may be taken into account when solving

the Lagrangean subproblem. Note that this aggregated decomposition gives the same bound as the decomposition defined in Section 3.1. This follows from the fact that the bound obtained from Lagrangean relaxation never exceeds the one obtained from a corresponding aggregated Lagrangean decomposition (see, for example, Guignard [8]), and that $Z(LE) = Z(DE)$.

4.3 Linking Subproblems

Linking subproblems are derived from Lagrangean decompositions and relaxations with respect to the flow conservation constraints. Note that we only have to consider two types of linking subproblems, since the weak and the strong formulations lead to the same problem structure.

This particular structure takes the following form: $Z(L_1) = \min \sum_{(i,j) \in A} \sum_{p \in P} \tilde{c}_{ij}^p x_{ij}^p + \sum_{(i,j) \in A} \tilde{f}_{ij} y_{ij}$, subject to constraints 7, 11, 6, 8, 9 and 10, where the vectors \tilde{c} and \tilde{f} represent, respectively, the routing and the fixed costs modified by the addition of the appropriate Lagrangean multipliers. This problem can be reformulated as $Z(L_1) = \min \sum_{(i,j) \in A} d_{ij} y_{ij}$, subject to constraints 8, 9 and 10, where, for each arc (i, j) , d_{ij} represents the optimal value of the following problem:

$$d_{ij} = \min \sum_{p \in P} \tilde{c}_{ij}^p x_{ij}^p + \tilde{f}_{ij} \quad (28)$$

$$\sum_{p \in P} x_{ij}^p \leq u_{ij} \quad (29)$$

$$0 \leq x_{ij}^p \leq b_{ij}^p \quad \forall p \in P \quad (30)$$

This continuous knapsack problem is easily solved by sorting the routing costs, considering only those that are negative. Given the optimal values of these continuous knapsack subproblems, a formulation that depends only on the design variables is obtained that, unless valid inequalities derived from the knapsack constraints are added, is solvable by inspection.

The linking subproblem derived from the extended formulation can be formulated as follows: $Z(L_2) = \min \sum_{(i,j) \in A} \sum_{p \in P} \sum_{k \in O(p)} \sum_{l \in D(p)} \tilde{c}_{ij}^{pkl} w_{ij}^{pkl} + \sum_{(i,j) \in A} \tilde{f}_{ij} y_{ij}$, subject to

constraints 19, 20, 21, 22, 18, 8, 9 and 10, where \tilde{c} and \tilde{f} are defined as above. An equivalent form is: $Z(L_2) = \min \sum_{(i,j) \in A} e_{ij} y_{ij}$, subject to constraints 8, 9 and 10, where for each arc (i, j) , e_{ij} is the optimal value of the following problem:

$$e_{ij} = \min \sum_{p \in P} \sum_{k \in O(p)} \sum_{l \in D(p)} \tilde{c}_{ij}^{pkl} w_{ij}^{pkl} + \tilde{f}_{ij} \quad (31)$$

$$\sum_{p \in P} \sum_{k \in O(p)} \sum_{l \in D(p)} w_{ij}^{pkl} \leq u_{ij} \quad (32)$$

$$\sum_{k \in O(p)} \sum_{l \in D(p)} w_{ij}^{pkl} \leq b_{ij}^p \quad \forall p \in P \quad (33)$$

$$\sum_{l \in D(p)} w_{ij}^{pkl} \leq \min(b_{ij}^p, o_k^p) \quad \forall p \in P, k \in O(p) \quad (34)$$

$$\sum_{k \in O(p)} w_{ij}^{pkl} \leq \min(b_{ij}^p, d_l^p) \quad \forall p \in P, l \in D(p) \quad (35)$$

$$w_{ij}^{pkl} \geq 0 \quad \forall p \in P, k \in O(p), l \in D(p) \quad (36)$$

Although at first sight, this subproblem appears more complicated than a continuous knapsack formulation, it can also be solved by sorting the routing costs. Once the optimal values have been determined, one again obtains a problem solvable by inspection, unless valid inequalities derived from the knapsack constraints are added to the formulation.

4.4 Knapsack Subproblems

The addition of all knapsack inequalities to any of the subproblems obtained by relaxation implies the resolution of 0-1 linear programming problems. Hence, it does not appear to be a computationally viable option. We note, in particular, that strong knapsack inequalities do not decompose easily, since the same design variable may appear in several constraints.

In the following, we define two sets of valid inequalities derived only from the weak knapsack constraints. Using any of these two sets of inequalities, we obtain subproblems that decompose into 0-1 knapsack problems, which are much easier to solve than general 0-1 linear programming problems (see Martello and Toth [20], for a survey of methods and computer codes for solving 0-1 knapsack problems).

The first set is obtained by combining the weak knapsack constraints 23, which decompose by origins, to the following inequalities, derived from relations 24:

$$\sum_{i \in N^-(j)-O} u_{ij} y_{ij} \geq \sum_{p \in P^D(j)} d_j^p - \sum_{i \in N^-(j) \cap O} u_{ij} \quad \forall j \in D \quad (37)$$

Similarly, the second set of inequalities is obtained by combining the weak knapsack constraints 24, which decompose by destinations, to the following inequalities, obtained from 23:

$$\sum_{j \in N^+(i)-D} u_{ij} y_{ij} \geq \sum_{p \in P^O(i)} o_i^p - \sum_{j \in N^+(i) \cap D} u_{ij} \quad \forall i \in O \quad (38)$$

It is easy to see that the addition of any of these two sets of inequalities to either the flow subproblem or the linking subproblem implies the resolution of $|O| + |D|$ 0-1 knapsack problems. To determine which set of inequalities should be added to produce the best bound, we suggest the following simple heuristic rule: if $|O| \geq |D|$, use the first set of inequalities, otherwise, use the second set.

4.5 Resource-Decomposition Heuristic

To compute upper bounds on the optimal value of the multicommodity capacitated network design problem, we consider the following multicommodity network flow problem, which is defined relative to a subset \bar{A} of A :

$$Z(M) = \min \sum_{(i,j) \in \bar{A}} \sum_{p \in P} \bar{c}_{ij}^p x_{ij}^p \quad (39)$$

$$\sum_{j \in \bar{N}^+(i)} x_{ij}^p - \sum_{j \in \bar{N}^-(i)} x_{ji}^p = \begin{cases} o_i^p & \text{if } i \in O(p) \\ -d_i^p & \text{if } i \in D(p) \\ 0 & \text{if } i \in T(p) \end{cases} \quad \forall i \in N, p \in P \quad (40)$$

$$\sum_{p \in P} x_{ij}^p \leq u_{ij} \quad \forall (i, j) \in \bar{A} \quad (41)$$

$$x_{ij}^p \leq b_{ij}^p \quad \forall (i, j) \in \bar{A}, p \in P \quad (42)$$

$$x_{ij}^p \geq 0 \quad \forall (i, j) \in \bar{A}, p \in P \quad (43)$$

In this formulation, for every node i , the sets of outward and inward neighbors, with respect to \bar{A} , are denoted, respectively, $\bar{N}^+(i)$ and $\bar{N}^-(i)$.

In order to approximate more closely the optimal value of the network design problem, we use a cost vector \bar{c} , which represents the routing costs modified to take into account the effect of the fixed costs. One formula that was shown to be useful in numerical experiments is the following:

$$\bar{c}_{ij}^p = c_{ij}^p + \frac{f_{ij}}{u_{ij}} \quad \forall (i, j) \in \bar{A}, p \in P \quad (44)$$

Another possibility is offered when lower bounds are computed by using Lagrangean relaxation with respect to the linking constraints. It then appears computationally attractive to adjust the routing costs by using the current values of the Lagrangean multipliers. Note, however, that empirically this strategy was not superior to the one defined by formula 44.

Given a feasible solution \bar{x} to the multicommodity network flow problem, an upper bound on the optimal value of the network design problem is given by

$$Z_u = \sum_{(i,j) \in \bar{A}} \sum_{p \in P} c_{ij}^p \bar{x}_{ij}^p + \sum_{(i,j) \in \bar{A}} f_{ij} \bar{y}_{ij} \quad (45)$$

where

$$\bar{y}_{ij} = \begin{cases} 0, & \text{if } \bar{x}_{ij}^p = 0, \forall p \in P \\ 1, & \text{otherwise} \end{cases} \quad \forall (i, j) \in \bar{A} \quad (46)$$

To compute feasible solutions to the multicommodity network flow problem, we reformulate it as a bilevel program:

$$Z(M) = \min z_u(v) \quad (47)$$

$$\sum_{p \in P} v_{ij}^p = u_{ij} \quad \forall (i, j) \in \bar{A} \quad (48)$$

$$v_{ij}^p \leq b_{ij}^p \quad \forall (i, j) \in \bar{A}, p \in P \quad (49)$$

$$v_{ij}^p \geq 0 \quad \forall (i, j) \in \bar{A}, p \in P \quad (50)$$

where $z_u(v)$ is the optimal value of the following *resource-decomposition subproblem*, which decomposes into $|P|$ single-commodity network flow problems:

$$z_u(v) = \min \sum_{(i,j) \in \bar{A}} \sum_{p \in P} \bar{c}_{ij}^p x_{ij}^p \quad (51)$$

subject to constraints 40, 43 and

$$x_{ij}^p \leq v_{ij}^p \quad \forall (i, j) \in \bar{A}, p \in P \quad (52)$$

Thus, given a point \bar{v} , the computational effort of finding a feasible solution involves projecting this point on the set C , defined by relations 48 to 50, and then solving $|P|$ single-commodity network flow problems. The projection can be accomplished by solving $|\bar{A}|$ singly-constrained quadratic programming problems (see Kennington and Helgason [12] for an efficient resolution procedure).

To obtain feasible solutions that approximate the optimal value of the multicommodity network flow problem, we use subgradient optimization in the following way. If we already have a feasible solution, we modify the current value of v by moving along the direction of a subgradient of $z_u(v)$. This defines a new point \bar{v} , which is then projected over C and used in the computation of the single-commodity network flow problems. If we don't have a feasible solution (e.g., initially, or when the current value of v does not allow to obtain a feasible solution), we determine a new point \bar{v} by finding an optimal solution to the following *price-decomposition subproblem*, obtained from Lagrangean relaxation with respect to the total capacity constraints 41:

$$z_l(\gamma) = \min \sum_{(i,j) \in \bar{A}} \sum_{p \in P} (\bar{c}_{ij}^p + \gamma_{ij}) x_{ij}^p - \sum_{(i,j) \in \bar{A}} \gamma_{ij} u_{ij} \quad (53)$$

subject to constraints 40, 42 and 43. The nonnegative Lagrangean vector γ is initially set to 0 and subsequently adjusted by moving along the direction of a subgradient of $z_l(\gamma)$.

In summary, given a subset \bar{A} of A and a current best known upper bound Z_u^* on the optimal value of the network design problem, the resource-decomposition heuristic is stated as follows:

1. Initialize the Lagrangean vector: $\gamma \leftarrow 0$. Initialize the iteration counters: $i \leftarrow 1$, $j \leftarrow 0$.
2. Solve $z_l(\gamma)$: if there is no feasible solution, stop. Otherwise, let \bar{v} be an optimal solution, and s_l a subgradient of $z_l(\gamma)$.

3. Increment j and project \bar{v} over C ; let v be the solution.
4. Solve $z_u(v)$: if there is no feasible solution, go to 7. Otherwise, let \bar{x} be an optimal solution, and s_u a subgradient of $z_u(v)$.
5. Compute an upper bound Z_u by using relations 45 and 46. Update the best known upper bound: $Z_u^* \leftarrow \min(Z_u^*, Z_u)$.
6. If $j \geq j_{\max}$, stop. Otherwise, update v by moving along the direction of s_u : $\bar{v} \leftarrow v + \theta_u s_u$. Go to 3.
7. If $j \geq j_{\max}$ or $i \geq i_{\max}$, stop. Otherwise, increment i and update γ by moving along the direction of s_l : $\gamma \leftarrow \gamma + \theta_l s_l$.
8. Solve $z_l(\gamma)$: let \bar{v} be an optimal solution, and s_l a subgradient of $z_l(\gamma)$. Go to 3.

Note that two iteration counters are used to evaluate properly the total computational effort. One counter measures the number of resource-decomposition subproblems that are solved, while the other counts the number of price-decomposition subproblems considered during the process.

When updating v in step 6, we use the following well-known formula for the stepsize: $\theta_u = \lambda_u (z_u(v) - Z_l(M)) / \|s_u\|^2$, where λ_u is a scalar, usually taken in the interval $(0, 2]$, $Z_l(M)$ is a lower bound on $Z(M)$, and $\|\cdot\|$ denotes the Euclidean norm. In our experiments, reported in Section 5, we use $\lambda_u = 1$ and set $Z_l(M) = z_l(0)$, the optimal value of the first solved price-decomposition subproblem.

When updating γ in step 7, we use the stepsize: $\theta_l = \lambda_l (Z_u(M) - z_l(\gamma)) / \|s_l\|^2$. In our experiments, we use $\lambda_l = 1$ and, initially, when no upper bound on $Z(M)$ is known, we set $Z_u(M) = 2 \times z_l(0)$. The possibility that this value does not represent a valid upper bound on $Z(M)$ is very unlikely, since this would mean that the gap between $z_l(0)$ and $Z(M)$ exceeds 100%. Once a first upper bound is found, we set Z_u to this value.

Note that we do not use a procedure to adjust the values of the scalars λ_u and λ_l in the course of the process, since, typically, the heuristic is executed for only a few iterations (around 10).

For a different implementation of a resource-decomposition heuristic based on subgradient optimization, see Kennington and Shalaby [13].

4.6 Bounding Procedures

In this subsection, we present a *composite procedure* that combines the upper bounding resource-decomposition heuristic to a *lower bounding procedure* that uses subgradient optimization to approximate the optimal value of any of the Lagrangean duals defined in Section 3.

We note μ the vector of Lagrangean multipliers associated with a given Lagrangean dual, and $z_d(\mu)$ the optimal value of the corresponding Lagrangean subproblem. At each step of the lower bounding procedure, $z_d(\mu)$ is determined, along with a subgradient s_d , and the current values of the multipliers are adjusted by moving θ_d along the direction of s_d , where $\theta_d = \lambda_d (Z_u - z_d(\mu)) / \|s_d\|^2$. Here, Z_u is a known upper bound on the optimal value of the network design problem (we discuss later the situation when such a bound is not yet determined). Since our objective is to approximate as closely as possible the optimal value of the Lagrangean dual, we allow the scalar λ_d to vary. To determine an appropriate adjustment of λ_d , we count the number of consecutive steps without improvement in the lower bound. When this number exceeds a given value, we halve the current value of λ_d (unless the result is smaller than a threshold ϵ_λ ; we then set λ_d to ϵ_λ to avoid numerical problems).

The lower bounding procedure can be executed in alternation with the resource-decomposition heuristic (see the description of the composite procedure below). It is stated as follows, assuming known values for μ , Z_l^* , the best known lower bound, and l^* , the number of consecutive steps (in previous executions of the procedure) without improvement in the lower bound:

1. Initialize the iteration counter: $l \leftarrow 1$.
2. Solve the Lagrangean subproblem. Let $z_d(\mu)$ be the optimal value and s_d be a subgradient.

3. If $z_d(\mu) > Z_l^*$, $Z_l^* \leftarrow z_d(\mu)$ and $l^* \leftarrow 0$. Go to 5.
4. Increment l^* . If $l^* \geq l_{\max}^*$, update the parameter λ_d and reinitialize l^* : $\lambda_d \leftarrow \max\left(\epsilon_\lambda, \frac{\lambda_d}{2}\right)$ and $l^* \leftarrow 0$.
5. Update μ by moving along the direction of s_d : $\mu \leftarrow \mu + \theta_d s_d$.
6. If $l \geq l_{\max}$, stop. Otherwise, increment l and go to 2.

The composite procedure runs as follows. It first applies the resource-decomposition heuristic with $\bar{A} = A$. This normally identifies an upper bound Z_u that can be used in computing the stepsizes in the lower bounding procedure. If this is not the case, we replace Z_u in the stepsize formula by $3 \times z_l(0)$, that is, we take three times the value of the first solved price-decomposition subproblem as an approximation to the optimal value of the network design problem. The event that this measure does not represent a valid upper bound is very unlikely, since this would mean that the gap between $z_l(0)$ and the optimal value of the network design problem exceeds 200%. The composite procedure then alternates between executions of the lower bounding procedure and the resource-decomposition heuristic, the solution of the lower bounding procedure being used to identify subsets \bar{A} . Specific rules to generate these subsets are discussed in Section 5. The steps of the procedure are stated as follows:

1. Initialize the iteration counter: $k \leftarrow 1$. Initialize the variables of the resource-decomposition heuristic: $\bar{A} \leftarrow A$ and $Z_u^* \leftarrow +\infty$. Apply the resource-decomposition heuristic. If there is no feasible solution to the first price-decomposition subproblem, stop. Otherwise, initialize the variables of the lower bounding procedure: $\mu \leftarrow 0$, $Z_l^* \leftarrow 0$ and $l^* \leftarrow 0$.
2. If $k \geq k_{\max}$, stop. Otherwise, increment k .
3. Apply the lower bounding procedure.

4. Generate one or more subsets \bar{A} by using the solution obtained at the last iteration of the lower bounding procedure. For each of these subsets (unless $\bar{A} = A$), apply the resource-decomposition heuristic. Go to 2.

5 Experimental Results

In this section, we report and analyze numerical results obtained by applying the bounding procedures described previously to a large set of test problems. The experiments were not intended to be exhaustive nor definitive, but mainly to provide insight about the quality of the bounds generated by the composite procedure. We have focused on the implementation of efficient bounding procedures based on the strong formulation only, and did not perform any experiment based on the extended formulation. We conjecture, however, that the main conclusions obtained by this computational study may be generalized to the extended formulation.

There are three main questions that we aim to address by these computational experiments:

- How the three types of Lagrangean methods (relaxation with respect to the linking constraints, relaxation with respect to the flow conservation constraints, decomposition) behave numerically?
- What is the improvement that one may obtain by using the strong formulation of the problem instead of the weak one?
- What is the improvement that one may obtain when valid inequalities derived from knapsack constraints are added to the strong formulation?

In order to answer these questions, we generated a large set of test problems, consisting of three types of networks: complete bipartite networks, general networks with single origin-single destination (single O-D) per commodity, and general networks with multiple origins and multiple destinations (multiple O-D) per commodity.

Bipartite and general networks are generated by using two different codes. To generate bipartite networks, one has to specify the number of origins, the number of destinations,

the number of commodities, as well as intervals for the uniform distribution that provides costs and capacities. The code used to obtain general networks includes a similar procedure to randomly generate costs and capacities. In addition, one has to specify the number of nodes, the number of arcs, and the number of commodities. Note that arcs are created by connecting two randomly selected nodes and that no parallel arcs are allowed. The number of origins (or destinations) per commodity is also randomly chosen in a given interval.

To obtain networks with various degrees of capacity, total capacities are scaled by using a measure called *capacity ratio*: $C = \sum_{(i,j) \in A} m_{ij} / \sum_{(i,j) \in A} u_{ij}$. In this formula, m_{ij} represents an upper bound on the amount of flow that can move through an arc (i, j) . For bipartite networks, this bound is set to $m_{ij} = \min(\sum_{p \in P} o_i^p, \sum_{p \in P} d_j^p)$, while for general networks, we use $m_{ij} = \sum_{p \in P} t_p$. When C approaches 1, the problem is lightly capacitated, while as C increases, it becomes more and more capacitated. For each test problem, the capacity ratio is adjusted in order to ensure that the problem is both feasible and capacitated.

We have not generated true partial capacities. For bipartite networks, we set the partial capacities to $b_{ij}^p = \min(u_{ij}, o_i^p, d_j^p) \quad \forall (i, j) \in A, p \in P$, while, for general networks, we use $b_{ij}^p = \min(u_{ij}, t_p) \quad \forall (i, j) \in A, p \in P$.

Similarly to capacities, fixed costs are scaled in order to generate problems where their relative importance varies significantly. For that purpose, we use the *fixed cost ratio*: $F = |P| \sum_{(i,j) \in A} f_{ij} / \sum_{(i,j) \in A} u_{ij} \sum_{p \in P} c_{ij}^p$. When F is close to 0, the fixed costs are low compared to the routing costs, while their relative importance increases proportionally with F .

We generated 32 instances of each of the three types of networks. In this Section, we present the results obtained on 18 problems, which are representative of the behavior of the entire set of test problems. For a more complete set of results, the reader is referred to our companion paper [5]. The characteristics of the 18 selected test problems are summarized in Table 1. A low value for C or F is represented by the letter L, while a high value is described by the letter H. The first six problems are bipartite networks,

Problem	$ N $	$ A $	$ P $	$ O(p) $	$ D(p) $	C	F
P1	50	400	10	40	10	L	L
P2	50	400	10	40	10	H	H
P3	50	400	100	40	10	L	L
P4	50	400	100	40	10	H	H
P5	100	1600	10	80	20	L	L
P6	100	1600	10	80	20	H	H
P7	20	230	40	1	1	L	L
P8	20	230	40	1	1	H	H
P9	20	230	200	1	1	L	L
P10	20	230	200	1	1	H	H
P11	30	520	100	1	1	L	L
P12	30	520	100	1	1	H	H
P13	20	230	40	10	10	L	L
P14	20	230	40	10	10	H	H
P15	20	230	200	10	10	L	L
P16	20	230	200	10	10	H	H
P17	30	520	100	15	15	L	L
P18	30	520	100	15	15	H	H

Table 1: Characteristics of test problems

problems P7 to P12 are single O-D networks, and the last six problems are multiple O-D networks.

The composite procedure is implemented in Fortran 77. The implementation allows to approximate, via the lower bounding procedure, the optimal values of the following Lagrangean duals: FS , LS , DS , FS_{K^*} , LS_{K^*} , $DS_{K^*}(L)$, $DS_{K^*}(F, L)$, where K^* is defined according to the ideas presented in Section 4.4.

In our implementation, the flow subproblems, which arise from applying Lagrangean decomposition or Lagrangean relaxation with respect to the linking constraints, are solved by removing the total capacity constraints, but by keeping the partial capacity constraints. As mentioned in Section 4.2, this choice leads essentially to the resolution of $|P|$ single-commodity network flow problems. For that purpose, we use a modified version of the primal simplex code RNET (version 3.61), due to Grigoriadis and Hsu [7]. The modifications were required in order to handle real costs and capacities, and to allow fast

reoptimizations by storing the solutions of all the $|P|$ single-commodity flow problems. The same code is used to solve the single-commodity network flow problems that arise when the resource-decomposition heuristic is applied. In order to solve 0-1 knapsack problems, we use the code MT1R, due to Martello and Toth [20].

The optimal values of the continuous relaxations of both the weak and the strong formulations are determined by using the commercial software CPLEX (version 2.1) [3]. The continuous relaxation of the weak formulation is reformulated as a multicommodity network flow problem, as suggested in section 4.1. Both problems are solved by using the “netopt” option.

All experiments are conducted on Sparc10/30 workstations. The implementation of the composite procedure is compiled with the *f77* compiler, using the *-O* option.

The parameters that limit the number of iterations in the resource-decomposition heuristic are given the values $j_{\max} = 10$ and $i_{\max} = 5$. The parameters of the lower bounding procedure are adjusted as follows. For all problems, we set $l_{\max}^* = 20$ and $\epsilon_\lambda = 0.00001$. When testing problems P1 to P16, we use, initially, $\lambda = 1$, while for problems P17 and P18, we use, initially, $\lambda = 0.1$. Although there may exist better adjustments, these values were found to be generally quite effective and representative of the performances of each method.

Table 2 displays the results of computing exactly, by using CPLEX, or approximately, by using the lower bounding procedure, the optimal values of the following problems: *CW*, *CS*, *FS*, *LS*, *DS*, *FS_{K*}*, *LS_{K*}*. We do not include the results obtained by approximating the optimal values of problems $DS_{K^*}(L)$ and $DS_{K^*}(F, L)$ since, in these cases, the lower bounding procedure behaves similarly to the case when the optimal value of *DS* is looked for. The composite procedure is executed with $k_{\max} = 2$ and $l_{\max} = 500$, except for problems P17 and P18, where we use $l_{\max} = 1000$ (the details of the upper bound computation are described below). For each problem, two figures are shown in each column: the first represents the value of the lower bound, rounded in hundreds of units, while the second indicates the time (CPU seconds) required to obtain this value.

With regard to the lower bounds obtained by the various procedures, we can draw

Problem	CW	CS	FS	LS	DS	FS_{K^*}	LS_{K^*}
P1	1794	2022	2017	1681	1464	2019	1766
	2	90	29	2	24	29	4
P2	3170	3275	3271	2369	1613	3320	2978
	11	265	37	2	32	36	5
P3	2183	2944	2773	2108	1225	2764	2219
	22	48952	576	45	430	560	47
P4	2944	3435	3335	2296	1201	3377	2659
	69	53785	478	45	421	549	47
P5	1950	2206	2198	1618	1292	2198	1700
	11	2208	183	7	120	185	15
P6	3692	3810	3805	2716	1278	3987	3356
	117	5256	173	7	163	206	21
P7	4266	4806	4803	4447	4064	4802	4450
	1	9	49	3	50	50	4
P8	5269	5808	5797	5037	4555	5788	5100
	1	20	50	3	49	54	4
P9	739	1097	1056	443	384	1054	536
	21	12153	311	40	273	314	42
P10	987	1356	1322	432	367	1332	632
	20	14051	294	39	270	304	41
P11	476	645	637	416	378	637	435
	22	1743	312	36	290	335	38
P12	742	961	947	418	371	944	498
	23	9446	335	36	304	351	39
P13	3095	3101	3091	2497	2490	3096	2708
	4	59	50	3	49	52	5
P14	4836	4841	4824	2427	2342	4821	3592
	3	216	51	3	52	55	5
P15	3766	3768	3765	2168	2185	3763	2697
	22	6168	275	40	268	281	42
P16	4879	4884	4878	2133	2152	4868	3367
	25	9481	278	40	276	286	42
P17	1227	1227	1203	919	929	1184	1005
	25	2896	577	72	560	609	78
P18	1856	1856	1784	832	828	1726	1264
	29	11000	592	72	565	628	79

Table 2: Lower bounds and computation times

the following conclusions:

- The lower bounding procedure approximates closely the optimal value of problem FS . We note also that the time to obtain this value generally dominates the one taken by CPLEX to compute $Z(CS)$, particularly when the number of commodities is high.
- The lower bounding procedure experiments difficulties in finding the optimal values of problems LS , DS and LS_{K^*} . To eliminate the possibility that this behavior is caused by an improper parameter adjustment, we tried many parameter settings: no significant improvements in the lower bound were, however, been found. The explanation rather lies in the “zigzag” phenomenon that the classical subgradient method may sometimes exhibit (see, for example, Lemaréchal [14]). When testing the method, we observed, in fact, that the bounds show significant up and down movements, without approaching the optimal value closely enough. As the value of the parameter λ is decreased, the bounds also display a tendency to zigzag around a value far from the optimal one.
- For bipartite and single O-D problems, there is a significant improvement when the strong formulation is used instead of the weak one. This is not the case, however, for multiple O-D problems where the improvement is very small. It is noteworthy that the strong and the extended formulations differ only for this class of problems; this is the only case where we could therefore expect an improvement if the extended formulations were used.
- There is small improvement in the lower bounds when knapsack inequalities are added to the strong formulation, as illustrated by the results obtained by approximating $Z(FS_{K^*})$. Note that the improvement is most significant for problems with high fixed costs (such as P2 and P6). Note also that the computational overhead of having to solve additional 0-1 knapsack problems is not significant and is sometimes compensated by a better convergence behavior (see, in particular, the results of problem P2).

To compute upper bounds, we describe two procedures to generate subsets \bar{A} . The first, called *primal procedure*, can be used with all Lagrangean methods: given an optimal solution to a Lagrangean subproblem, \bar{A} is obtained by removing from A the arcs for which all related flow and design variables have value 0. The second, called *dual procedure*, can be used only with methods FS and FS_{K^*} . It makes use of the *reduced fixed costs*, defined for each arc (i, j) as: $r_{ij} = f_{ij} - \alpha_{ij}u_{ij} - \sum_{p \in P} \beta_{ij}^p b_{ij}^p$, where α_{ij} and β_{ij}^p are the Lagrangean multipliers associated, respectively, to each weak and strong linking constraint. The reduced fixed costs are first sorted in increasing order. Then, \bar{A} is obtained by removing half of the arcs with the highest reduced fixed costs. After applying the resource-decomposition heuristic, a new subset \bar{A} is defined as follows. If a better upper bound has been found, half of the remaining arcs with the highest fixed costs are removed. Otherwise, half of the arcs that were removed previously are added. This dichotomic scheme is continued until no arcs can be added or removed without changing the current \bar{A} .

Table 3 shows the gaps, measured as $100 \times (Z_u^* - Z_l^*)/Z_l^*$, and the upper bound computation times (CPU seconds) when using the primal and dual procedures in conjunction with the computation of $Z(FS)$. The results of the primal procedure are obtained with $k_{\max} = 11$ and $l_{\max} = 50$, except for problems P17 and P18 where $l_{\max} = 100$ is used. Thus, approximately the same lower bounds as the ones displayed in Table 2 are computed, since the total number of subgradient iterations is the same. The gaps obtained by the dual procedure are based on the results shown in Table 2.

Results indicate that the dual procedure improves significantly over the primal procedure. Other results, reported in [5], also show that the upper bounds are generally better when used in conjunction with the lower bounding procedure to approximate $Z(FS)$. Therefore, our general conclusion is that the best Lagrangean method, among those that were tested, consists in relaxing the linking constraints, and that the addition of knapsack inequalities does not improve significantly the lower bounds.

Problem	Primal(FS)		Dual(FS)	
	gap	time	gap	time
P1	12.17	24	6.02	14
P2	30.37	14	45.49	15
P3	86.79	378	69.71	186
P4	82.08	365	42.58	154
P5	25.35	95	25.60	60
P6	52.93	59	49.60	87
P7	36.11	46	2.62	15
P8	57.38	45	11.46	13
P9	220.86	207	121.84	121
P10	223.26	270	121.76	110
P11	141.97	297	37.23	124
P12	245.58	295	179.73	163
P13	61.52	62	55.54	31
P14	124.65	60	84.12	29
P15	124.57	365	96.68	141
P16	142.90	358	94.23	141
P17	75.01	267	54.98	208
P18	166.77	313	85.83	198

Table 3: Gaps and upper bound computation times

6 Conclusions and Perspectives

In this paper, we have presented three formulations of a general multicommodity capacitated network design problem. The three formulations define a hierarchy of continuous relaxations. We showed how these continuous relaxations can be approximated efficiently by using three types of Lagrangean methods. Moreover, we demonstrated that the lower bounds can be improved by adding valid knapsack inequalities. We also described an efficient procedure, based on the resource-decomposition principle, for identifying feasible solutions of good quality.

We presented results of preliminary computational experiments that focus on one of the formulations of the problem. Following our computational study, some general conclusions can be drawn:

- Lagrangean decomposition and Lagrangean relaxation with respect to the flow con-

servation constraints have nice practical and theoretical properties. However, we experimented difficulties in evaluating the optimal value of their associated Lagrangean duals. If these relaxation methods are to be used, a more sophisticated method than the classical subgradient procedure should be called upon in order to adjust properly the Lagrangean multipliers.

- The best performances are obtained by relaxing the linking constraints. However, the gaps shown by running the composite procedure, are still very large, particularly for problems with high fixed costs. To improve the lower bounds, a better understanding of the polyhedral structure of the problem is required. A natural extension to the present work would consist in relaxing, in a Lagrangean sense, valid inequalities added to the formulations (for examples of such approaches for solving difficult combinatorial problems, see Fischer [4], and Lucena [15]).
- For multiple O-D problems, efficient procedures based on the extended formulation could improve the lower bounds. We suggest, in Sections 4.2 and 4.3, some possible avenues for such developments.
- To improve the upper bounds determined by the composite procedure, we may combine the resource-decomposition heuristic to classical local search methods, or to procedures based on Tabu Search principles.
- The computational effort required by the composite procedure may be significantly reduced if parallel implementations are designed to exploit the inherent decomposition properties of the procedure.

Acknowledgments

This research has been supported by grants from the Fonds F.C.A.R. of the Province of Québec, and the Natural Sciences and Engineering Research Council of Canada. We also want to acknowledge the efforts of Mr. Benoit Bourbeau who helped us with the experimentations.

References

- [1] Ahuja R.K., Magnanti T.L. and Orlin J.B. (1993), *Network Flows: Theory, Algorithms, and Applications*, Prentice-Hall.
- [2] Cornuejols G., Sridharan R. and Thizy J.M. (1991), *A Comparison of Heuristics and Relaxations for the Capacitated Plant Location Problem*, European Journal of Operational Research, 50, 280-297.
- [3] CPLEX (1993), *CPLEX Documentation*, CPLEX Optimization, Inc.
- [4] Fischer M.L. (1990), *Optimal Solution of Vehicle Routing Problems Using Minimum k -Trees*, Technical Report, Department of Decision Sciences, The Wharton School, University of Pennsylvania.
- [5] Gendron B. and Crainic T.G. (1994), *Experiments on Multicommodity Capacitated Network Design Problems*, Working Paper, Centre de recherche sur les transports, Université de Montréal.
- [6] Geoffrion A.M. (1974), *Lagrangian Relaxation for Integer Programming*, Mathematical Programming Study, 2, 82-114.
- [7] Grigoriadis M.D. and Hsu T. (1979), *RNET 3.61 Documentation*, Rutgers University.
- [8] Guignard M. (1990), *General Aggregation Schemes in Lagrangian Decomposition: Theory and Potential Applications*, Working Paper, Department of Decision Sciences, The Wharton School, University of Pennsylvania.
- [9] Guignard M. and Kim S. (1987), *Lagrangian Decomposition: A Model Yielding Stronger Lagrangian Bounds*, Mathematical Programming, 39, 215-228.
- [10] Helgason R.V. (1980), *A Lagrangian Relaxation Approach to the Generalized Fixed Charge Network Flow Problem*, Ph.D. Thesis, Southern Methodist University.

- [11] Hurd J.K. and Murphy F.H. (1992), *Exploiting Special Structure in Primal Dual Interior Point Methods*, ORSA Journal on Computing, 4, 38-44.
- [12] Kennington J.L. and Helgason R.V. (1980), *Algorithms for Network Programming*, Wiley.
- [13] Kennington J.L. and Shalaby M. (1978), *An Effective Subgradient Procedure for Minimal Cost Multicommodity Flow Problems*, Management Science, 23, 994-1004.
- [14] Lemaréchal C. (1989), *Nondifferentiable Optimization*, in *Handbooks in Operations Research and Management Science*, Volume 1, *Optimization*, G.L. Nemhauser, A.H.G. Rinnooy Kan and M.J. Todd Editors, 529-572, North-Holland.
- [15] Lucena A. (1993), *Steiner Problem in Graphs: Lagrangean Relaxation and Cutting-Planes*, presented at NETFLOW93, San Miniato, Italy, October 3-7 (see Technical Report TR-21/93, Dipartimento di Informatica, Università degli Studi di Pisa, 147-154).
- [16] Magnanti T.L. (1993), *Modeling and Solving Network Design Problems*, presented at NETFLOW93, San Miniato, Italy, October 3-7 (see Technical Report TR-21/93, Dipartimento di Informatica, Università degli Studi di Pisa, 155-159).
- [17] Magnanti T.L., Mirchandani P. and Vachani R. (1991), *Modeling and Solving the Capacitated Network Loading Problem*, Working Paper No. 709, Katz Graduate School of Business, University of Pittsburgh.
- [18] Magnanti T.L., Mirchandani P. and Vachani R. (1993), *The Convex Hull of Two Core Capacitated Network Design Problems*, Mathematical Programming, 60, 233-250.
- [19] Magnanti T.L. and Wong R.T. (1984), *Network Design and Transportation Planning: Models and Algorithms*, Transportation Science, 18(1), 1-55.
- [20] Martello S. and Toth P. (1990), *Knapsack Problems: Algorithms and Computer Implementations*, Wiley.

- [21] Minoux M. (1989), *Network Synthesis and Optimum Network Design Problems: Models, Solution Methods and Applications*, Networks, 19, 313-360.
- [22] Morris J.G. (1978), *On the Extent to which Certain Fixed-Charge Depot Location Problems can be Solved by LP*, Journal of the Operational Research Society, 29, 71-76.
- [23] Nauss R.M. (1978), *An Improved Algorithm for the Capacitated Facility Location Problem*, Journal of the Operational Research Society, 29(12), 1195-1201.
- [24] Padberg M.W., Van Roy T.J. and Wolsey L.A. (1985), *Valid Linear Inequalities for Fixed Charge Problems*, Operations Research, 33, 842-861.
- [25] Rardin R.L. (1982), *Tight Relaxations of Fixed Charge Network Flow Problems*, Report J-82-3, School of industrial Engineering, Purdue University.
- [26] Rardin R.L. and Choe U. (1979), *Tighter Relaxations of Fixed Charge Network Flow Problems*, Report J-79-18, School of Industrial and Systems Engineering, Georgia Institute of Technology.
- [27] Schrage L. (1975), *Implicit Representation of Variable Upper Bounds in Linear Programming*, Mathematical Programming Study, 4, 113-138.
- [28] Todd M.J. (1982), *An Implementation of the Simplex Method for Linear Programming Problems with Variable Upper Bounds*, Mathematical Programming, 23, 34-49.
- [29] Todd M.J. (1988), *Exploiting Special Structure in Karmarker's Linear Programming Algorithm*, Mathematical Programming, 41, 81-103.
- [30] Van Roy T.J. and Wolsey L.A. (1987), *Solving Mixed Integer Programming Problems Using Automatic Reformulation*, Operations Research, 35, 45-57.