

**BOUNDING PROCEDURES FOR MULTICOMMODITY
CAPACITATED FIXED CHARGE NETWORK DESIGN
PROBLEMS**

Bernard Gendron

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

January 1996

Abstract

This paper presents bounding procedures, based on Lagrangean relaxation and resource-decomposition, for solving multicommodity capacitated fixed charge network design problems, both with and without integrality restrictions on the flow variables. Computational experiments on a large set of randomly generated test problems demonstrate that these bounding procedures generally outperform a state-of-the-art LP/MIP solver, used interactively with the best available options for our class of problems. In particular, these experiments show that our procedures are well-suited for instances with a very large number of commodities.

Key words : Multicommodity capacitated fixed charge network design, Lagrangean relaxation, resource-decomposition.

Résumé

Dans cet article, nous présentons des procédures de calcul de bornes, basées sur la relaxation lagrangienne et la décomposition selon les ressources, pour résoudre des problèmes de conception de réseaux multiproduit avec capacités et coûts fixes. Nous considérons deux variantes du problème: l'une où les flots peuvent être fractionnaires et l'autre où les flots doivent être entiers. Des résultats d'essais numériques sur un grand ensemble de problèmes générés aléatoirement démontrent la supériorité de ces procédures, lorsqu'elles sont comparées à un logiciel de programmation linéaire de haute performance, utilisé de manière interactive avec les meilleures options disponibles pour notre classe de problèmes. En particulier, les essais numériques établissent l'efficacité de nos procédures à résoudre des problèmes où le nombre de produits est très grand.

Mots-clés : Conception de réseau multiproduit avec capacités et coûts fixes, relaxation lagrangienne, décomposition selon les ressources.

1 Introduction

Network design models have been known for long as useful planning tools in areas such as transportation, telecommunications, manufacturing and logistics, among others. Comprehensive surveys on the applications of network design models and their resolution by mathematical programming techniques can be found in Magnanti and Wong [16], Minoux [17], and Magnanti [15]. As pointed out by the last author, *multicommodity capacitated* versions of a general *fixed charge network design model* pose considerable algorithmic challenges (see Rardin and Choe [20], Rardin [19], Helgason [11], and Balakrishnan [2], for studies of related formulations and some previous attempts at solving these difficult problems). The present paper can be seen as a step towards the efficient resolution of large-scale multicommodity capacitated fixed charge network design problems. Its main objective is to present and study bounding procedures based on Lagrangean relaxation and resource-decomposition.

Among the main features and novelties of the paper, we may cite:

- The proposed procedures are easily adapted to the difficult case where integrality restrictions on the flow variables are imposed. We present computational results obtained by our procedures for that case and compare them with the results obtained by a traditional LP-based branch-and-bound code implemented in a state-of-the-art LP/MIP solver.
- Computational experiments demonstrate that the Lagrangean relaxation approach outperforms, in terms of solution times, a state-of-the-art LP solver that uses a specialized simplex-based network optimizer to solve the continuous relaxation. The experiments also show that the Lagrangean relaxation approach is especially well-suited for instances with a very large number of commodities.
- The upper bounding procedure, based on resource-decomposition, makes use of the solutions generated by the Lagrangean subproblems in a novel fashion to evade the difficulty of finding feasible solutions in multicommodity networks. Several capacity

allocations are generated by projecting the flows obtained from solving Lagrangean subproblems. For each capacity allocation, a generalized add-drop heuristic is used, that adds and drops subsets of arcs according to a criterion based on the values of the flows and of the multipliers obtained from the Lagrangean subproblem. This procedure is quite effective, in terms of solution quality, for problems where the flow costs dominate the fixed costs. Although not as effective for other classes of problems, the procedure is still quite efficient, in terms of solution times, especially when compared to a traditional LP-based branch-and-bound code implemented in a state-of-the-art LP/MIP solver.

The paper is organized as follows. In Section 2, we formulate the problem and describe the bounding procedures. Section 3 presents and analyzes computational results on a large set of randomly generated test problems. In Section 4, we summarize our main results and discuss possible extensions to this work.

2 Formulation and Bounding Procedures

We consider 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* σ_i^p , and to each destination $i \in D(p)$ a positive *demand* d_i^p . We assume that $\sum_{i \in O(p)} \sigma_i^p = \sum_{i \in D(p)} d_i^p = t_p$ 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) . We also assume that all quantities defined above are integers. A nonnegative *flow 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 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

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

We also introduce the sets of outward and inward neighbors, respectively, 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 problem may then be formulated as

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

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

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

$$x_{ij}^p \geq 0 \quad \forall (i, j) \in A, p \in P \quad (5)$$

$$\sum_{p \in P} x_{ij}^p \leq u_{ij} y_{ij} \quad \forall (i, j) \in A \quad (6)$$

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

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

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

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

In addition to this basic model, we also consider a version of the problem where integrality restrictions are imposed on the flow variables:

$$x_{ij}^p \text{ integer} \quad \forall (i, j) \in A, p \in P \quad (11)$$

Our bounding procedures are easily adapted to this difficult case. Indeed, the lower bounding procedure is identical, while the upper bounding procedure only requires a slight modification to handle these integrality constraints. Note that even when the partial capacity constraints 4 are redundant, our experience has shown that their inclusion into the model generally accelerates the convergence of relaxation methods (LP-based simplex approach or Lagrangean relaxation with subgradient optimization). Note also that constraints 7 are redundant, but their addition to the model considerably improves the lower bounds obtained through relaxations (see Gendron and Crainic [6], and also Cornuéjols, Sridharan and Thizy [3], for a similar observation for the capacitated plant location problem).

2.1 Lower Bounding Procedure

To compute lower bounds, we propose an approach based on relaxing constraints 6 and 7 [6]. The resulting Lagrangean subproblem takes the form:

$$Z(\alpha, \beta) = \min \sum_{p \in P} \sum_{(i,j) \in A} (c_{ij}^p + \alpha_{ij} + \beta_{ij}^p) x_{ij}^p + \sum_{(i,j) \in A} \left(f_{ij} - \alpha_{ij} u_{ij} - \sum_{p \in P} \beta_{ij}^p b_{ij}^p \right) y_{ij} \quad (12)$$

subject to 3, 4, 5, 8, 9 and 10, where α and β are nonnegative Lagrangean multipliers associated to constraints 6 and 7, respectively. This problem decomposes into two parts. The first part, which depends only on the flow variables, may be decomposed into $|P|$ single-commodity minimum cost network flow problems, for which adding integrality restrictions does not affect their optimal values (see, for example, Ahuja, Magnanti and Orlin [1]). The second part, which depends only on the design variables, may be solved easily, without considering the integrality constraints on the design variables, and its optimal solution \bar{y} is given by

$$\bar{y}_{ij} = \begin{cases} 0, & \text{if } r_{ij} > 0 \\ 1, & \text{otherwise} \end{cases} \quad \forall (i, j) \in A \quad (13)$$

where $r_{ij} = f_{ij} - \alpha_{ij} u_{ij} - \sum_{p \in P} \beta_{ij}^p b_{ij}^p$ is the *reduced fixed cost* associated to arc (i, j) .

Since the Lagrangean subproblem can be solved without any integrality constraint, it has the Integrality property of Geoffrion [8]. Thus, the best lower bound one can

hope for when using this Lagrangean relaxation is equal to the optimal value of the continuous relaxation, obtained by dropping the integrality constraints. In Section 3, however, we report computational experiments showing that the Lagrangean relaxation approach outperforms a state-of-the-art LP solver that uses a specialized simplex-based network optimizer to solve the continuous relaxation.

As is well-known (see, for example, Nemhauser and Wolsey [18]), the Lagrangean dual, considered as a function of α and β , is concave, but nondifferentiable. To maximize it, and consequently derive the best Lagrangean lower bound, we use an adaptation of the subgradient method, based on the early works of Held, Wolfe and Crowder [10], and Crowder [4] (for a recent survey of nondifferentiable optimization techniques, see Lemaréchal [13]). Given an optimal solution (\bar{x}, \bar{y}) to the Lagrangean subproblem, a subgradient $(s(\alpha), s(\beta))$ of the Lagrangean dual at (α, β) is given by

$$s_{ij}(\alpha) = \sum_{p \in P} \bar{x}_{ij}^p - u_{ij} \bar{y}_{ij} \quad \forall (i, j) \in A \quad (14)$$

$$s_{ij}^p(\beta) = \bar{x}_{ij}^p - b_{ij}^p \bar{y}_{ij} \quad \forall (i, j) \in A, p \in P \quad (15)$$

At iteration $T \geq 1$, the Lagrangean subproblem is solved and the multipliers are adjusted by moving $\rho(\alpha, \beta)$ along a direction $(d^{(T)}(\alpha), d^{(T)}(\beta))$, defined as

$$d_{ij}^{(T)}(\alpha) = s_{ij}(\alpha) + \theta d_{ij}^{(T-1)}(\alpha) \quad \forall (i, j) \in A \quad (16)$$

$$d_{ij}^{p(T)}(\beta) = s_{ij}^p(\alpha) + \theta d_{ij}^{p(T-1)}(\beta) \quad \forall (i, j) \in A, p \in P \quad (17)$$

where $(d^{(0)}(\alpha), d^{(0)}(\beta)) = (0, 0)$ and $0 \leq \theta < 1$. Given this direction, the multipliers are adjusted as

$$\alpha_{ij}^{(T)} = \max(0, \alpha_{ij}^{(T-1)} + \rho(\alpha, \beta) d_{ij}^{(T)}(\alpha)) \quad \forall (i, j) \in A \quad (18)$$

$$\beta_{ij}^{p(T)} = \max(0, \beta_{ij}^{p(T-1)} + \rho(\alpha, \beta) d_{ij}^{p(T)}(\beta)) \quad \forall (i, j) \in A, p \in P \quad (19)$$

The stepsize is computed as $\rho(\alpha, \beta) = \lambda(\bar{Z}^l - Z(\alpha, \beta)) / \|(s(\alpha), s(\beta))\|^2$, where $0 < \lambda \leq 2$, \bar{Z}^l is an estimate of the optimal value of the Lagrangean dual, and $\|\cdot\|$ denotes the Euclidean norm. The estimate is evaluated as $\bar{Z}^l = \min(2X\tilde{Z}^l, \tilde{Z}^u)$, where \tilde{Z}^l and \tilde{Z}^u are, respectively, the best known lower and upper bounds on the optimal value of the

network design problem. To approximate as closely as possible the optimal value of the Lagrangean dual, we allow the scalar λ to vary. To determine an appropriate adjustment of λ , 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 (unless the result is smaller than a threshold, fixed at 0.001, when we set λ to this value to avoid numerical problems).

The lower bounding procedure may be performed in alternation with the upper bounding procedure, to be described in Section 2.2. Two iteration counters are used to provide stopping criteria. A local counter t , reinitialized every time the procedure is called, stops the procedure when it reaches a given value t_{max} . A global counter T , used in computing the directions given by 16 and 17, stops the procedure and the algorithm when it reaches a given value T_{max} .

Given known values, either initialized or determined by the previous run of the procedure, for λ and θ , the multipliers α and β , \tilde{Z}^l , the best known lower bound, T^* , the number of consecutive steps without improvement in the lower bound, T , the total number of iterations and $(d^{(T)}(\alpha), d^{(T)}(\beta))$, the previous direction, the lower bounding procedure may be stated as follows:

1. Initialize iteration counter: $t \leftarrow 0$.
2. $t \leftarrow t + 1$; $T \leftarrow T + 1$.
3. Solve the Lagrangean subproblem to obtain (\bar{x}, \bar{y}) , $Z(\alpha, \beta)$ and $(s(\alpha), s(\beta))$.
4. If $Z(\alpha, \beta) > \tilde{Z}^l$, $\tilde{Z}^l \leftarrow Z(\alpha, \beta)$ and $T^* \leftarrow 0$; otherwise, $T^* \leftarrow T^* + 1$.
5. If $T \geq T_{max}$, stop.
6. If $T^* \geq T_{max}^*$, $\lambda \leftarrow \max(0.001, \lambda/2)$ and $T^* \leftarrow 0$.
7. Compute $(d^{(T)}(\alpha), d^{(T)}(\beta))$ and $\rho(\alpha, \beta)$; update (α, β) .
8. If $t \geq t_{max}$, stop; otherwise, increment t and go to 2.

2.2 Upper Bounding Procedure

The upper bounding procedure, based on the resource-decomposition principle, makes use of the optimal solution (\bar{x}, \bar{y}) of the last Lagrangean subproblem. It proceeds as follows. First, allocated partial capacities are obtained by solving the following *projection problem*:

$$\min \sum_{(i,j) \in A} \sum_{p \in P} \left(x_{ij}^p - \bar{x}_{ij}^p \right)^2 \quad (20)$$

subject to constraints 4, 5 and

$$\sum_{p \in P} x_{ij}^p = u_{ij} \quad \forall (i, j) \in A \quad (21)$$

This problem decomposes into $|A|$ singly-constrained quadratic programming problems (see Kennington and Helgason [12], for an efficient resolution procedure). If we consider the version of the problem where integrality restrictions are imposed on the flow variables, we can obtain integral allocated capacities simply by rounding down to the nearest integers the fractional components of the optimal solution to the projection problem.

Denote v the resulting allocated capacities. We then solve the following *resource-decomposition subproblem*, which decomposes into $|P|$ single-commodity minimum cost network flow problems:

$$Z^u(v) = \min \sum_{p \in P} \sum_{(i,j) \in A} \left(c_{ij}^p + f_{ij}/u_{ij} \right) x_{ij}^p \quad (22)$$

subject to constraints 3, 5 and

$$x_{ij}^p \leq v_{ij}^p \quad \forall (i, j) \in A, p \in P \quad (23)$$

Given an optimal solution to this problem, an upper bound on the optimal value of the network design problem is obtained by evaluating the objective function in 2 at this point, where the values of the design variables are given by definition 1.

If there is no feasible solution, given the current allocated capacities, the procedure stops. Otherwise, it tries to improve the current solution by using a *generalized add-drop heuristic* that works as follows. First, the arcs are sorted in increasing order of the

following criterion: $q_{ij} = \omega \sum_{p \in P} \bar{x}_{ij}^p + r_{ij}$, where ω is an arbitrarily large negative number. This criterion is based on the rationale that the arcs with the smallest (respectively largest) q_{ij} are more (respectively less) likely to appear in an optimal solution. Then, arcs are dropped or added following a *dichotomic scheme* that makes use of the sorting criterion to identify the most interesting arcs. At each step, a subset \bar{A} of A is defined and the resource-decomposition subproblem, restricted to \bar{A} , is solved. The dichotomic scheme starts by dropping half of the arcs, those with the largest q_{ij} , to obtain \bar{A} . Following the resolution of the resource-decomposition subproblem, if a feasible solution is obtained, the second half of the arcs in \bar{A} , those with the largest q_{ij} in \bar{A} , are dropped from \bar{A} . Otherwise, if no feasible solution is found, the first half of the arcs not in \bar{A} , those with the smallest q_{ij} in A/\bar{A} , are added to \bar{A} . The process continues until it is no longer possible to add or remove arcs without obtaining a set already considered.

Given a current best known upper bound \tilde{Z}^u , the upper bounding procedure may be stated as follows:

1. Using the optimal solution to the last Lagrangean subproblem, solve the projection problem. If integrality restrictions on flow variables are imposed, round down the fractional components of the optimal solution to the nearest integers. Let v be the resulting allocated capacities.
2. Solve the resource-decomposition subproblem.
3. If there is no feasible solution, stop.
4. Compute an upper bound Z^u ; if $Z^u < \tilde{Z}^u$, then $\tilde{Z}^u \leftarrow Z^u$.
5. Sort the arcs in increasing order of $q_{ij} = \omega \sum_{p \in P} \bar{x}_{ij}^p + r_{ij}$.
6. Initialize variables of the dichotomic scheme: $p^l \leftarrow 1$, $p^u \leftarrow |A|$.
7. If $p^u - p^l \leq 1$, stop.
8. Define \bar{A} as the first $p^l + (p^u - p^l)/2$ sorted arcs.

9. Solve the resource-decomposition subproblem.
10. If there is no feasible solution, then $p^l \leftarrow p^l + (p^u - p^l)/2$ (add); go to 7.
11. Compute an upper bound Z^u ; if $Z^u < \tilde{Z}^u$, then $\tilde{Z}^u \leftarrow Z^u$.
12. $p^u \leftarrow p^l + (p^u - p^l)/2$ (drop); go to 7.

To this basic procedure, we also tried to incorporate three features that could improve its performance. The first is to perturb locally the allocated capacities. One way to perform this task is by moving along the direction of a subgradient of $Z^u(v)$, considered as a function of v . However, results were generally disappointing due to the difficulty of identifying feasible allocations. The second feature is to modify the objective function in the resource-decomposition subproblem. In particular, we tried to add the current values of the Lagrangean multipliers, and a penalty term that favors variables with large quantities of flows in the optimal solution of the last Lagrangean subproblem. Both modifications generally did not produce any improvement. The third feature is to follow the generalized add-drop heuristic by a generalized swap heuristic. Again, this was rather disappointing, due to the fact that the sorting criterion is quite effective in identifying interesting arcs, so that swapping subsets of arcs generally gives no improvement.

2.3 Composite Procedure

To combine efficiently the two bounding procedures, we perform a composite procedure that consists of two phases. The first phase is the pure *relaxation phase*, where the lower bounding procedure is applied for up to \tilde{t}_{max} iterations. Then, some parameters of the lower bounding procedure may be reset, and we enter in the *relaxation/decomposition* phase, where the lower and upper bounding procedures are performed alternatively. The rationale behind this division into two phases comes from the experimental observation that it is generally useless to apply the upper bounding procedure at the early stages, when the optimal solution of the Lagrangean subproblem does not provide a good approximation.

The composite procedure may be stated as follows:

1. Initialize parameters: $\tilde{t}_{max}, T_{max}, t_{max}^*, \lambda, \theta$.
2. Initialize variables of the upper bounding procedure: $\tilde{Z}^u \leftarrow +\infty$; and variables of the lower bounding procedure: $(\alpha, \beta) \leftarrow (0, 0)$, $(d^{(0)}(\alpha), d^{(0)}(\beta)) = (0, 0)$, $\tilde{Z}^l \leftarrow 0$, $t^* \leftarrow 0$, $T \leftarrow 0$.
3. (Relaxation phase) Apply lower bounding procedure with $t_{max} = \tilde{t}_{max}$.
4. Reset parameters λ, θ, t_{max} .
5. (Relaxation/decomposition phase) Apply lower bounding procedure.
6. If $T \geq T_{max}$, stop.
7. Apply upper bounding procedure; go to 5.

3 Experimental Results

In this section, we report and analyze numerical results obtained by applying the composite procedure to a large set of test problems, which consists of two types of networks: complete bipartite networks and general networks with single origin-single destination per commodity. Furthermore, we divide general networks into two subclasses: asymmetric problems, for which flow costs on each arc differ by commodity, and symmetric problems, for which every arc has the same flow cost for all commodities. 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. To obtain networks with various degrees of capacity, total capacities are scaled by using the *capacity ratio* $C = \sum_{(i,j) \in A} m_{ij} / \sum_{(i,j) \in A} u_{ij}$, where m_{ij} is an upper bound on the amount

of flow that can move through 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. In addition, for bipartite networks, we set the partial capacities to $b_{ij}^p = \min(u_{ij}, o_i^p, d_j^p)$, while, for general networks, we use $b_{ij}^p = \min(u_{ij}, t_p)$. 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 . In this section, we present the results obtained on 24 problems, which are representative of the behavior of a larger set of test problems. The characteristics of the 24 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.

In our Fortran 77 implementation, the minimum cost network flow problems are solved by using a modified version of the primal simplex code RNET (version 3.61) [9]. The modifications are 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.

We compare our bounding procedures to the state-of-the-art LP/MIP solver CPLEX (version 2.1), used interactively with the best available options for our class of problems. For solving the continuous relaxations, we use the “netopt” option, which first applies a specialized simplex-based network optimizer followed by dual simplex iterations to treat the additional constraints. For solving the two versions of the problems where flow variables are continuous and integers, respectively, we use the “optimize” option, which performs a primal simplex-based branch-and-bound method. We set a CPU time limit of 24 hours and, for the branch-and-bound method, a limit of 20000 on the number of nodes and a tree memory limit of 64 megabytes.

All experiments are conducted on Sparc10/514 workstations, each equipped with a memory of 64 megabytes. The code is compiled with the *f77* compiler, using the -

| problem | $ N $ | $ A $ | $ P $ | $ O(p) $ | $ D(p) $ | C | F | structure |
|---------|-------|-------|-------|----------|----------|-----|-----|------------|
| P1 | 50 | 400 | 10 | 40 | 10 | L | L | bipartite |
| P2 | 50 | 400 | 10 | 40 | 10 | H | H | bipartite |
| P3 | 50 | 400 | 100 | 40 | 10 | L | L | bipartite |
| P4 | 50 | 400 | 100 | 40 | 10 | H | H | bipartite |
| P5 | 50 | 625 | 100 | 25 | 25 | L | H | bipartite |
| P6 | 50 | 625 | 100 | 25 | 25 | H | L | bipartite |
| P7 | 100 | 1600 | 10 | 80 | 20 | L | L | bipartite |
| P8 | 100 | 1600 | 10 | 80 | 20 | H | H | bipartite |
| P9 | 20 | 230 | 40 | 1 | 1 | L | L | asymmetric |
| P10 | 20 | 230 | 40 | 1 | 1 | H | H | asymmetric |
| P11 | 20 | 230 | 200 | 1 | 1 | L | L | asymmetric |
| P12 | 20 | 230 | 200 | 1 | 1 | H | H | asymmetric |
| P13 | 20 | 300 | 200 | 1 | 1 | L | H | asymmetric |
| P14 | 20 | 300 | 200 | 1 | 1 | H | L | asymmetric |
| P15 | 30 | 520 | 100 | 1 | 1 | L | L | asymmetric |
| P16 | 30 | 520 | 100 | 1 | 1 | H | H | asymmetric |
| P17 | 25 | 100 | 10 | 1 | 1 | H | H | symmetric |
| P18 | 25 | 100 | 30 | 1 | 1 | H | H | symmetric |
| P19 | 100 | 400 | 10 | 1 | 1 | H | H | symmetric |
| P20 | 20 | 230 | 40 | 1 | 1 | H | H | symmetric |
| P21 | 20 | 300 | 40 | 1 | 1 | H | H | symmetric |
| P22 | 20 | 300 | 200 | 1 | 1 | H | H | symmetric |
| P23 | 30 | 520 | 100 | 1 | 1 | H | H | symmetric |
| P24 | 30 | 700 | 100 | 1 | 1 | H | H | symmetric |

Table 1: Characteristics of Test Problems

O option. For all test problems, we use $\lambda = 2$, initially, $\theta = 0.9$ and $T_{\max}^* = 10$. The relaxation phase is applied for $\tilde{t}_{\max} = 450$ lower bound iterations, and then the parameter λ , which usually attains the threshold value 0.001 at this stage, is reset to 0.01, in order to subsequently generate Lagrangean multipliers that vary significantly from one iteration to the next. In the relaxation/decomposition phase, the upper bounding procedure is applied after every lower bound iteration. A total of $T_{\max} = 500$ lower bound iterations are performed in the two phases.

Tables 2, 3 and 4 compare the results of the bounding procedures to those obtained by using CPLEX. In each table and for each problem, we show the values of the bounds obtained and the CPU times, in seconds, necessary to compute them. In Tables 3 and 4, an “X” indicates that no feasible solution was found.

From these figures, we can draw the following main conclusions:

- The Lagrangean relaxation approach provides very good approximations to the optimal values of the continuous relaxations. Moreover, it is generally much more efficient, in terms of solution times, than CPLEX “netopt”, especially for problems with a large number of commodities.
- For the variant with continuous flow variables, the resource-decomposition heuristic identifies feasible solutions to all of the test problems within a reasonable amount of time. By comparison, when used with the given time and memory constraints, CPLEX “optimize” is able to find feasible solutions to only 13 of the test problems, while only five of these problems are solved to optimality. We observe that the heuristic generally comes close to an optimal solution for problems where the flow costs dominate the fixed costs (such as P9 and P15). Although not as effective for other classes of problems, the heuristic is still quite efficient, in terms of solution times.
- The variant with integer flow variables is much more difficult to solve than its counterpart, even for the resource-decomposition heuristic. Indeed, using the heuristic, no feasible solution are found for five problems, all having tight capacity constraints,

| problem | Lagrangean relaxation | | CPLEX netopt | |
|---------|-----------------------|------|--------------|-------|
| | value | time | value | time |
| P1 | 202144 | 25 | 202196 | 75 |
| P2 | 327381 | 29 | 327538 | 200 |
| P3 | 292623 | 629 | 294400 | 48203 |
| P4 | 341596 | 262 | 343530 | 53957 |
| P5 | 344458 | 360 | 325913 t | 86523 |
| P6 | 201944 | 362 | 201107 t | 86472 |
| P7 | 220474 | 106 | 220597 | 2154 |
| P8 | 380852 | 110 | 380974 | 4761 |
| P9 | 480598 | 31 | 480599 | 6 |
| P10 | 580826 | 32 | 580827 | 10 |
| P11 | 108645 | 187 | 109694 | 10942 |
| P12 | 134546 | 180 | 135563 | 13094 |
| P13 | 122822 | 188 | 123545 | 19621 |
| P14 | 88911 | 188 | 89346 | 12960 |
| P15 | 64380 | 184 | 64519 | 1417 |
| P16 | 95644 | 167 | 96095 | 9168 |
| P17 | 43452 | 4 | 43455 | 5 |
| P18 | 82372 | 13 | 82419 | 26 |
| P19 | 48332 | 14 | 48375 | 166 |
| P20 | 633432 | 29 | 633466 | 14 |
| P21 | 596816 | 36 | 596839 | 23 |
| P22 | 103430 | 199 | 103633 | 12174 |
| P23 | 93690 | 150 | 94011 | 6784 |
| P24 | 53538 | 186 | 53661 | 3440 |

t: CPU time limit exceeded (24 hours)

Table 2: Comparison of Lower Bounding Procedure with CPLEX

| problem | resource decomposition | | CPLEX optimize | |
|---------|------------------------|------|----------------|-------|
| | value | time | value | time |
| P1 | 212008 | 26 | 206317 m | 11039 |
| P2 | 458372 | 42 | 340879 m | 18742 |
| P3 | 348035 | 395 | X t | 86448 |
| P4 | 433161 | 347 | X t | 86421 |
| P5 | 471414 | 492 | X t | 86441 |
| P6 | 433161 | 347 | X t | 86453 |
| P7 | 259220 | 133 | 228571 t | 86476 |
| P8 | 551263 | 199 | X t | 86402 |
| P9 | 483999 | 29 | 482333 | 48 |
| P10 | 598123 | 33 | 588334 | 1067 |
| P11 | 166017 | 136 | X t | 87064 |
| P12 | 325121 | 72 | X t | 89860 |
| P13 | 347728 | 164 | X t | 91465 |
| P14 | 178509 | 68 | X t | 88537 |
| P15 | 66451 | 208 | 65284 m | 12548 |
| P16 | 295282 | 110 | X t | 86579 |
| P17 | 59020 | 4 | 49899 | 274 |
| P18 | 103185 | 10 | 86621 m | 4791 |
| P19 | 197741 | 19 | 67274 m | 15619 |
| P20 | 702040 | 14 | 643036 | 19692 |
| P21 | 749605 | 40 | 604198 | 869 |
| P22 | 325918 | 83 | X t | 86620 |
| P23 | 278684 | 101 | 98500 t | 86580 |
| P24 | 101537 | 210 | 56565 m | 77816 |

t: CPU time limit exceeded (24 hours)
m: Tree memory limit exceeded (64 megabytes)

Table 3: Comparison of Upper Bounding Procedure with CPLEX – Continuous Flow Variables

| problem | resource decomposition | | CPLEX optimize | |
|---------|------------------------|------|----------------|-------|
| | value | time | value | time |
| P1 | 213724 | 24 | 220944 m | 4972 |
| P2 | 483502 | 28 | 420426 m | 9246 |
| P3 | 348773 | 365 | X t | 86436 |
| P4 | 450221 | 300 | X t | 86436 |
| P5 | 492922 | 425 | X t | 86435 |
| P6 | 450221 | 300 | X t | 86410 |
| P7 | 290152 | 109 | X m | 42239 |
| P8 | 689757 | 168 | X t | 86430 |
| P9 | 483999 | 29 | 482333 | 43 |
| P10 | 598123 | 33 | 588334 | 1027 |
| P11 | 178947 | 97 | X t | 86588 |
| P12 | X | 50 | X t | 89097 |
| P13 | 481374 | 69 | X t | 92440 |
| P14 | X | 63 | X t | 88874 |
| P15 | 81066 | 189 | 65284 m | 12678 |
| P16 | X | 49 | X t | 86618 |
| P17 | 67675 | 3 | 51712 n | 3192 |
| P18 | 105618 | 6 | 92141 m | 2079 |
| P19 | 189449 | 17 | 118018 m | 9524 |
| P20 | 713769 | 13 | 643036 | 20177 |
| P21 | 761307 | 40 | 604198 | 849 |
| P22 | X | 61 | X t | 86704 |
| P23 | X | 45 | 105125 t | 87130 |
| P24 | 107510 | 131 | 56588 m | 77538 |

t: CPU time limit exceeded (24 hours)
m: Tree memory limit exceeded (64 megabytes)
n: Maximum number of nodes exceeded (20000 nodes)

Table 4: Comparison of Upper Bounding Procedure with CPLEX – Integer Flow Variables

while other problems show a decrease in solution quality, sometimes considerable. Finally, note that CPLEX “optimize” is able to find feasible solutions to only 12 of the test problems, while only four of these problems are solved to optimality.

4 Conclusion

We presented bounding procedures, based on Lagrangean relaxation and resource-decomposition, for solving multicommodity capacitated fixed-charge network design problems, both with and without integrality restrictions on the flow variables. Computational experiments on a set of randomly generated test problems demonstrated that these bounding procedures generally outperform the state-of-the-art LP/MIP solver CPLEX, used interactively with the best available options for our class of problems. In particular, these experiments showed that our procedures are well-suited for instances with a large number of commodities.

As promising extensions to the present work, we suggest the following:

- The lower bounds can be further improved by adding strong valid inequalities to the model (as evidenced by the computational experiments, there are instances showing a strong integrality gap). Relaxing these inequalities in a Lagrangean fashion appears especially promising, since the Lagrangean subproblem would exhibit the same structure (for examples of such approaches for solving difficult combinatorial problems, see Fisher [5], and Lucena [14]).
- The resource-decomposition heuristic identifies many feasible solutions, while requiring a low computational effort. Thus, it may be used to provide initial solutions to more sophisticated heuristics.
- The computational effort required by the procedures may be significantly reduced by designing parallel implementations that exploit a decomposition by commodity [7]. Such implementations would be especially profitable for problems with a large number of commodities.

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 Benoit Bourbeau and Alexandre Lebouthilier 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] Balakrishnan A. (1984), *Valid Inequalities for the Network Design Problem with an Application to LTL Consolidation Problem*, Ph.D. Thesis, Sloan School of Management, Massachusetts Institute of Technology, Cambridge.
- [3] Cornuéjols 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.
- [4] Crowder H. (1976), *Computational Improvements for Subgradient Optimization*, Symposia Mathematica, Vol. XIX, 357-372, Academic Press, London.
- [5] Fisher M.L. (1994), *Optimal Solution of Vehicle Routing Problems Using Minimum K-Trees*, Operations Research 42, 393-410.
- [6] Gendron B. and Crainic T.G. (1994), *Relaxations for Multicommodity Capacitated Network Design Problems*, Publication CRT-965, Centre de recherche sur les transports, Université de Montréal.
- [7] Gendron B. and Crainic T.G. (1994), *Parallel Implementations of Bounding Procedures for Multicommodity Capacitated Network Design Problems*, Publication CRT-94-45, Centre de recherche sur les transports, Université de Montréal.

- [8] Geoffrion A.M. (1974), *Lagrangean Relaxation for Integer Programming*, Mathematical Programming Study, 2, 82-114.
- [9] Grigoriadis M.D. and Hsu T. (1979), *RNET 3.61 Documentation*, Rutgers University.
- [10] Held M., Wolfe P. and Crowder H.P. (1974), *Validation of Subgradient Optimization*, Mathematical Programming, 6, 62-88.
- [11] Helgason R.V. (1980), *A Lagrangean Relaxation Approach to the Generalized Fixed Charge Network Flow Problem*, Ph.D. Thesis, Southern Methodist University.
- [12] Kennington J.L. and Helgason R.V. (1980), *Algorithms for Network Programming*, Wiley.
- [13] 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.
- [14] 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).
- [15] 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).
- [16] Magnanti T.L. and Wong R.T. (1984), *Network Design and Transportation Planning: Models and Algorithms*, Transportation Science, 18(1), 1-55.
- [17] Minoux M. (1989), *Network Synthesis and Optimum Network Design Problems: Models, Solution Methods and Applications*, Networks, 19, 313-360.

- [18] Nemhauser G.L. and Wolsey L.A. (1988), *Integer and Combinatorial Optimization*, Wiley-Interscience.
- [19] Rardin R.L. (1982), *Tight Relaxations of Fixed Charge Network Flow Problems*, Report J-82-3, School of Industrial Engineering, Purdue University.
- [20] 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, Purdue University.