
Computational Optimization and Applications, 27, 187–215, 2004

An Augmented Lagrangian Algorithm for Large Scale Multicommodity Routing

TORBJÖRN LARSSON tolar@mai.liu.se
Department of Mathematics, Linköping Institute of Technology, SE-581 83 Linköping, Sweden

DI YUAN diyua@itn.liu.se
Department of Science and Technology, Linköping Institute of Technology, Campus Norrköping, SE-601 74 Norrköping, Sweden

Abstract. The linear multicommodity network flow (MCNF) problem has many applications in the areas of transportation and telecommunications. It has therefore received much attention, and many algorithms that exploit the problem structure have been suggested and implemented. The practical difficulty of solving MCNF models increases fast with respect to the problem size, and especially with respect to the number of commodities. Applications in telecommunications typically lead to instances with huge numbers of commodities, and tackling such instances computationally is challenging.

In this paper, we describe and evaluate a fast and convergent lower-bounding procedure which is based on an augmented Lagrangian reformulation of MCNF, that is, a combined Lagrangian relaxation and penalty approach. The algorithm is specially designed for solving very large scale MCNF instances. Compared to a standard Lagrangian relaxation approach, it has more favorable convergence characteristics. To solve the nonlinear augmented Lagrangian subproblem, we apply a disaggregate simplicial decomposition scheme, which fully exploits the structure of the subproblem and has good reoptimization capabilities. Finally, the augmented Lagrangian algorithm can also be used to provide heuristic upper bounds.

The efficiency of the augmented Lagrangian method is demonstrated through computational experiments on large scale instances. In particular, it provides near-optimal solutions to instances with over 3,600 nodes, 14,000 arcs and 80,000 commodities within reasonable computing time.

Keywords: multicommodity, augmented Lagrangian, routing

1. Introduction

In the linear multicommodity network flow (MCNF) problem, a number of different commodities are to be sent from origins to destinations in a network with mutual arc capacities, and at minimum cost. The MCNF model appears in a number of practical planning situations, and is a key component of many network design problems. Typical areas where MCNF models arise are transportation, production, and telecommunications systems. Examples of specific applications can be found in [1, 22, 26, 28, 39, 41].

In this paper, we consider applications of MCNF models in telecommunications networks. In these applications, a commodity is defined by a pair of nodes with a certain communication demand; such a node pair is referred to as an origin–destination (O–D) pair. A solution to the MCNF model provides a minimum cost routing plan, in which more than one route may be used to carry the flow of an O–D pair. (If the flow of an O–D pair has to follow

a single route, the problem is referred to as the integer multicommodity flow problem [7].)

The linear MCNF model is well-studied. (See e.g. [2, 45] for surveys of early developments.) The solution approaches vary from price- and resource-directive decomposition, partitioning methods, and dual ascent methods, to primal-dual heuristics. Recent approaches include the bundle method and interior point methods. Most research work on solution methods for this problem focuses on linear optimization techniques, see e.g. [16, 22, 31, 34, 42, 44, 54]. The MCNF model is a linear program; its practical difficulty stems from the fact that in a real application, the problem dimension tends to be huge. As the number of variables in an arc flow formulation equals the number of arcs times the number of commodities, an instance with 1,000 arcs and 1,000 commodities results in a model with 1 million variables. Message-routing problems in telecommunications applications often have large numbers of O–D pairs. For message-routing problems with a linear objective function, the largest instance presented and solved in the literature, to the best of our knowledge, has 500 nodes, 1300 arcs and 5,850 commodities, which gives an LP-model with approximately 7.6 million variables [5]. For message-routing problems with an average delay (nonlinear) objective function, instances with 106 nodes, 904 arcs, and 11,130 commodities (giving approximately 10 million variables) have recently been solved [58].

We are motivated to develop fast, approximate solution methods for the linear MCNF problem because of its applications in the area of telecommunications, where most instances are extremely large. We describe and evaluate a fast and convergent lower-bounding procedure which is based on an augmented Lagrangian reformulation of MCNF. When applied in a truncated fashion, the algorithm provides near-optimal solutions and tight lower bounds to the optimal value, at low computational effort and low memory requirement, even for large scale instances.

Augmented Lagrangian methods combine relaxation and penalty techniques. Compared to a traditional Lagrangian heuristic scheme, which has been applied to the MCNF problem in [38], an augmented Lagrangian approach often shows more favorable convergence characteristics, because the dual function is differentiable. Due to the nonlinearity of the penalty term, the augmented Lagrangian subproblem is nonlinear. The subproblem does not, however, need to be solved to optimality, especially not in the early stages of the algorithm, in order to ensure the global convergence. To solve the subproblem, we apply a disaggregate simplicial decomposition (DSD) method [48], which is very well adapted to the subproblem's structure. The DSD algorithm solves the subproblem by a decomposition based on linearization, and provides information to the search in the Lagrangian dual space.

The empirical efficiency of our augmented Lagrangian algorithm is demonstrated through computational experiments on large scale instances. The largest problem instance used in these experiments has 3,600 nodes, 14,160 arcs, and 80,000 commodities; the equivalent arc flow formulation has more than 1 billion variables. For this instance the algorithm is capable of providing near-optimal solutions with a relative accuracy of 0.2%.

The remainder of this paper is organized as follows. In Section 2 we present two mathematical formulations of the linear MCNF problem. Section 2 also includes a review of previously suggested solution methods. In Section 3 our augmented Lagrangian approach is outlined, and the details of the algorithm are discussed in Section 4. Computational results

are presented in Section 5, which also includes a computational study of some other codes for the problem. Conclusions are drawn in Section 6.

2. Problem description

2.1. *Mathematical formulations*

The network is denoted by $\mathcal{G} = (\mathcal{N}, \mathcal{A})$, where $\mathcal{N}$ and $\mathcal{A}$ are the sets of nodes and directed arcs, respectively. We use $\mathcal{A}_i^+$ and $\mathcal{A}_i^-$ to denote the sets of arcs that emanate and terminate, respectively, at node i. Let $\mathcal{K}$ be a set of commodities. Each commodity $k \in \mathcal{K}$ is characterized by a pair of nodes: its origin $o(k)$ and its destination $d(k)$. We use d_k to denote the amount of traffic to be routed from $o(k)$ to $d(k)$, that is, the demand of commodity k. Associated with each arc $a \in \mathcal{A}$ is a mutual (coupling) capacity u_a. Let further $c_{ka} \geq 0$ denote the unit cost for routing commodity k along arc a. In the remainder of the paper, we presume that there is a feasible routing solution with respect to the demands and the capacities.

Letting variable x_{ka} denote the flow of commodity k on arc a, the following arc flow formulation is obtained.

[MCNF–A]

$$v^* = \min \sum_{k\in\mathcal{K}} \sum_{a\in\mathcal{A}} c_{ka} x_{ka}$$

$$\text{s.t.} \sum_{a\in\mathcal{A}_i^+} x_{ka} - \sum_{a\in\mathcal{A}_i^-} x_{ka} = \begin{cases} d_k & \text{if } i = o(k) \\ -d_k & \text{if } i = d(k) \\ 0 & \text{otherwise} \end{cases} \quad \forall i \in \mathcal{N}, \quad \forall k \in \mathcal{K} \tag{1A}$$

$$\sum_{k\in\mathcal{K}} x_{ka} \leq u_a \qquad \forall a \in \mathcal{A} \tag{2A}$$

$$x_{ka} \geq 0 \qquad \forall a \in \mathcal{A}, \quad \forall k \in \mathcal{K}$$

Constraint sets (1A) and (2A) are flow conservation and mutual capacity constraints, respectively.

MCNF–A is a straightforward formulation and contains $|\mathcal{A}| \cdot |\mathcal{K}|$ variables. The problem size increases fast with respect to the size of the network and the number of commodities. If the routing cost c_{ka} is the same for all $k \in \mathcal{K}$, then the MCNF problem can alternatively be formulated in terms of origin-specific (or destination-specific) commodities, each of which is the group of O–D commodities that share one and same origin (or destination). This aggregation yields a model with considerably fewer variables and constraints. Some solution methods are however less efficient in practice for an aggregated formulation than for the disaggregated one. This is illustrated well by the experiments conducted in [42].

Another frequently used model for MCNF is the path flow formulation, in which arc flows are defined in terms of path flows. Letting $\mathcal{P}_k$ denote the set of elementary paths from

$o(k)$ to $d(k)$, and letting h_{kr} be the flow on path $r \in \mathcal{P}_k$, the path flow formulation is

$$
\begin{aligned}
[\textbf{MCNF–P}] \quad v^* = \min \sum_{k\in\mathcal{K}} \sum_{a\in\mathcal{A}} c_{ka} x_{ka} & \\
\text{s.t.} \quad \sum_{r\in\mathcal{P}_k} h_{kr} &= d_k \quad \forall k \in \mathcal{K} && \text{(1P)}\\
\sum_{r\in\mathcal{P}_k} \delta_{kra} h_{kr} &= x_{ka} \quad \forall a \in \mathcal{A},\ \forall k \in \mathcal{K} && \text{(2P)}\\
\sum_{k\in\mathcal{K}} x_{ka} &= x_a \quad \forall a \in \mathcal{A} && \text{(3P)}\\
x_a &\le u_a \quad \forall a \in \mathcal{A} && \text{(4P)}\\
h_{kr} &\ge 0 \quad \forall r \in \mathcal{P}_k,\ \forall k \in \mathcal{K}
\end{aligned}
$$

with

$$
\delta_{kra} = \begin{cases} 1 & \text{if path } r \in \mathcal{P}_k \text{ uses arc a,} \\ 0 & \text{otherwise} \end{cases}
$$

being the so called arc-path incidence matrix, and x_a being the total flow on arc a. In MCNF–P, (1P) and (4P) are the demand and the mutual capacity constraints, respectively. Constraint sets (2P) and (3P) establish the relationship between arc and path flows.

Note that MCNF–P can be stated in terms of the path flow variables only, by using the constraints (2P) and (3P) to eliminate the arc flow variables from the model. It is however advantageous, both conceptually and computationally, to keep the arc flow variables in the model, and to handle the constraints (2P) and (3P) implicitly.

The path flow formulation MCNF–P has significantly fewer constraints than the arc flow formulation MCNF–A, but the number of path flow variables is typically huge, because it grows exponentially with respect to the network size. In practice, it is therefore necessary to solve MCNF–P in a framework of column (path) generation, where a column is generated through the solution of a shortest path problem. This solution approach for MCNF–P can easily be shown [63] to be equivalent to applying the Dantzig–Wolfe decomposition, with a disaggregate master problem [42], to the arc flow formulation MCNF–A. An extreme point generated by the Dantzig–Wolfe subproblem for a single commodity corresponds to sending the entire demand along an elementary path.

The algorithm to be presented in this paper can, similarly to the above mentioned equivalence, be derived from either MCNF–P or MCNF–A. For the sake of convenience, we choose to base our presentation on the formulation MCNF–P.

2.2. A review of solution methods

A wide range of solution methods have been developed for MCNF problems. These methods can be divided into some categories.

A *price-directive decomposition* approach dualizes the mutual capacity constraints using Lagrangian multipliers (or dual variables, or prices), so that the subproblem separates by commodity. A straightforward approach for solving the dual problem is Dantzig–Wolfe

decomposition (e.g. [3, 18, 22, 42]). Alternative approaches for solving the dual problem are subgradient optimization [38], bundle methods [23, 25], and analytic center cutting plane methods [17]. The augmented Lagrangian method described in this paper is also a price-directive decomposition scheme.

Another category of methods comprises *resource-directive decomposition*, in which the right-hand-sides of the mutual capacity constraints are allocated among the commodities, in order to separate the problem by commodity. Subgradient optimization has been proposed by several authors [3, 19, 35, 46, 62] for finding the optimal capacity allocation.

Partitioning methods for linear programs operate with a partial explicit basis, while the remaining part of the basis is handled implicitly. This type of methods has been applied to MCNF–A in [1, 16, 31, 32, 34, 44, 53]. Partitioning techniques for MCNF–P can be found in [5, 20, 56]. Recent developments of partitioning methods for MCNF models can be found in [54] and [55].

Drawing much attention recently, *combinatorial approximation algorithms* comprise another approach to MCNF problems. The focus in this field lies on the development of MCNF algorithms with theoretically guaranteed computational complexity and error bounds, see, for example, [21, 43, 51]. In [29, 61] implementations of approximation algorithms are described, and shown to be efficient and competitive to some other approaches.

Other approaches that have been suggested for solving MCNF models include the Kornai–Lipták method [33], barrier/penalty methods [10, 30], and dual ascent [4], as well as primal-dual heuristics [6].

3. The augmented Lagrangian method

3.1. Motivation

The Dantzig–Wolfe decomposition approach has shown to be promising for the MCNF problem that we consider here (see for example [3, 17]), at least up to a moderate problem size. In Dantzig–Wolfe decomposition, a master program is used to optimally combine the previously generated extreme points, and to provide new dual prices to the subproblem. The method will finitely provide an optimal solution. An inherent weakness is that the master problem becomes computationally time-consuming for large scale instances. Indeed, we observe in our computational experiments (see Section 5) that for large instances, most time is spent on solving the master problem.

In contrast to the master problem in the Dantzig–Wolfe decomposition, one may use a simple subgradient search method to update the dual prices. In each step of this search method, a computationally inexpensive Lagrangian subproblem (which in our case coincides with the Dantzig–Wolfe subproblem) is solved. The solution to the subproblem produces a subgradient to the Lagrangian dual function, which is non-differentiable. A subgradient optimization procedure has asymptotic convergence in the dual space. However, such a method suffers from lack of coordinability, i.e., that the sequence of solutions to the Lagrangian subproblem does not, in general, converge to an optimal primal solution. (For a discussion about the non-coordinability phenomenon and some possible counter-measures, see e.g. [50].) Another weakness of subgradient optimization is that the dual convergence rate is typically poor.

An augmented Lagrangian method attempts to achieve a compromise between the above two approaches. The augmented Lagrangian dual function is differentiable, which implies coordinability, i.e., that a convergent dual search method will, in the limit, produce an optimal primal solution. The dual search is conducted by a simple scheme, and no master problem is needed. The computationally demanding part of an augmented Lagrangian method is the solution of the nonlinear subproblem. Convergence can however be obtained also in the case of inexact solution of the subproblem. In [49], an augmented Lagrangian algorithm was proposed for the capacitated traffic assignment problem, which is similar to the MCNF problem studied here.

Our augmented Lagrangian method is purely price-directive, and without any resource allocation. As will be more clear later, the decomposition in our algorithm is carried out at three levels. These properties make the algorithm different from, for example, the exponential multiplier approach [30], which first computes a strictly feasible solution, and then applies an exact penalty method where capacities are allocated among the commodities.

3.2. An outline of the augmented Lagrangian method

We outline here our augmented Lagrangian method for MCNF. For a general treatment of augmented Lagrangian approaches, the reader is referred to, for example, [8, 9, 47].

To simplify the presentation, we restate MCNF in the following form.

$$\begin{aligned} [\textbf{MCNF}] \quad v^* = \min \quad & f(x) \\ \text{s.t. } & g_a(x_a) \le 0 \quad \forall a \in \mathcal{A} \\ & x \in X \end{aligned}$$

Here, f represents the objective function in MCNF–P, $g_a(x_a) = x_a/u_a - 1$, and X is the set of arc flows that are feasible in the constraints (1P)–(3P) in MCNF–P.

In an exterior penalty method, some or all of the constraints of the original optimization model are included in the objective function, by means of a penalty term that imposes a (high) cost to infeasible solutions. An augmented Lagrangian scheme is a combined exterior penalty and relaxation method. Using a quadratic penalty function, the augmented Lagrangian function reads

$$L_c(x, \mu) = f(x) + \sum_{a \in \mathcal{A}} \mu_a g_a^+(x_a, \mu_a, c) + \sum_{a \in \mathcal{A}} \frac{c}{2} g_a^+(x_a, \mu_a, c)^2, \tag{1}$$

where $\mu_a \ge 0$, $a \in \mathcal{A}$ are the Lagrangian multipliers, $c > 0$ is a penalty parameter, and $g_a^+(x_a, \mu_a, c) = \max\{g_a(x), -\mu_a/c\}$. When c approaches zero, $L_c(x, \mu)$ approaches an ordinary Lagrangian function. The function $L_c(x, \mu)$ can alternatively be stated in the form

$$L_c(x, \mu) = f(x) + \frac{1}{2c} \sum_{a \in \mathcal{A}} \left\{ [\mu_a + c g_a(x_a)]_+^2 - \mu_a^2 \right\}, \tag{2}$$

where the operator $[\cdot]_+ = \max\{0, \cdot\}$. The augmented Lagrangian subproblem of our solution scheme is the nonlinear problem

$$\textbf{[SUB]} \quad L_c(\mu) = \min_{x \in X} L_c(x, \mu), \tag{3}$$

and a subproblem solution is given by

$$x(\mu, c) \in \arg\min_{x \in X} L_c(x, \mu). \tag{4}$$

For any $\mu \geq 0$, the value $L_c(\mu)$ is a lower bound to v^*, and, because MCNF is assumed to be feasible, we have for any positive c that

$$v^* = \max_{\mu \geq 0} L_c(\mu). \tag{5}$$

The dual function $L_c(\mu)$ is concave and differentiable (e.g. [9], p. 352), with

$$\frac{\partial L_c(\mu)}{\partial \mu_a} = g_a^+(x_a(\mu, c), \mu_a, c), \quad \forall\, a \in \mathcal{A}. \tag{6}$$

Minimization of $L_c(x, \mu)$ with respect to x produces near-optimal (but typically infeasible) solutions to MCNF when the Lagrangian multipliers are close to a dual optimal solution μ^*, or when the penalty parameter c is sufficiently large. Both mechanisms are utilized in an augmented Lagrangian approach. To achieve convergence to a dual optimal solution, an iterative search is performed. Letting μ^l be the dual iterate in iteration l, it can be shown [47] that the updating formula

$$\mu_a^{l+1} = \left[\mu_a^l + c g_a(x_a(\mu^l, c))\right]_+, \quad \forall\, a \in \mathcal{A} \tag{7}$$

ensures that $L_c(\mu^l) \to v^*$ and $\mu^l \to \mu^*$ when $l \to \infty$. The convergence analysis in [47] utilizes the fact that this formula gives a steepest ascent step on the differentiable dual function $L_c(\mu)$. To improve the convergence rate, it is a standard measure (e.g. [9], p. 346) to combine the search in the dual space with a non-decreasing sequence of penalty parameters $\{c^l\}$. It can be shown [47] that the above convergence results still hold if $c^0 > 0$ and $c^l \geq c^{l-1}$, $\forall\, l \geq 1$.

The practical convergence rate depends heavily on the choice of the sequence $\{c^l\}$. A large penalty value accelerates the convergence rate in the dual space, but imposes ill-conditioning in SUB. We use an update rule based on the infeasibility of the subproblem solution ([9], p. 347). The penalty parameter is increased by a constant factor $\beta_c > 1$ if the total infeasibility is not decreased sufficiently, say by a factor $\gamma < 1$, in the latest iteration. To avoid ill-conditioning in SUB, the value of the penalty parameter is not allowed to exceed an upper bound $c_{\max}$. (As established below, convergence is ensured for any choice of $c_{\max} > c^0$.) We hence set

$$c^{l+1} = \begin{cases} \min\{\beta_c c^l, c_{\max}\} & \text{if } \|g^+(x^l, \mu^l, c^l)\| > \gamma \|g^+(x^{l-1}, \mu^{l-1}, c^{l-1})\| \\ c^l & \text{otherwise,} \end{cases} \tag{8}$$

with $g^+(x, \mu, c) = [g_a^+(x_a, \mu_a, c)]_{a \in \mathcal{A}}$.

In practice, the solution method for SUB is usually terminated before an accurate approximation of $x(\mu^l, c^l)$ is found. With ε_L^l being the required relative accuracy in the lth iteration, the approximate subproblem solution, denoted x^l, is required to satisfy $L_{c^l}(x^l, \mu^l) \leq (1 + \varepsilon_L^l)L_{c^l}(\mu^l)$. One strategy to ensure convergence (e.g. [9], p. 347) is then to use a diminishing sequence of ε_L^l, such as, for example,

$$\varepsilon_L^{l+1} = \beta_L \cdot \varepsilon_L^l \tag{9}$$

with $\beta_L \in (0, 1)$.

The disaggregate simplicial decomposition algorithm, which is used to solve SUB (see Section 3.3), provides lower bounds to $L_{c^l}(\mu^l)$; these lower bounds are used, instead of the unknown value $L_{c^l}(\mu^l)$, in the termination criterion for SUB.

We outline the augmented Lagrangian algorithm below.

1. (*Initialization*). Set $l = 0$ and $\underline{v} = -\infty$. Select an initial dual solution $\mu^0 \geq 0$, a penalty parameter $c^0 > 0$, and an upper bound $c_{\max} > c^0$; select $\varepsilon_L^0 > 0$ and $\beta_L \in (0, 1)$.
2. (*Subproblem solution*). Solve the augmented Lagrangian subproblem, SUB, to the relative accuracy ε_L^l. Denote the solution by x^l, and let $\bar{v}_L^l = L_{c^l}(x^l, \mu^l)$ and $\underline{v}_L^l$ be the final upper and lower bounds, respectively, to $L_{c^l}(\mu^l)$. [The bounds satisfy $(\bar{v}_L^l - \underline{v}_L^l)/\underline{v}_L^l \leq \varepsilon_L^l$.] If $\underline{v}_L^l > \underline{v}$, set $\underline{v} = \underline{v}_L^l$.
3. (*Termination check*). If some predefined criteria are satisfied, terminate the algorithm and return $\underline{v}$ as the best known lower bound to v^*.
4. (*Value updates*). Let μ^{l+1} be given by (7), c^{l+1} by (8), and ε_L^{l+1} by (9). Set $l = l + 1$, go to Step 2.

The following proposition provides a convergence result for the above algorithm.

Proposition 1 (*Convergence result of the augmented Lagrangian scheme*)**.** *The sequences* $\{\mu^l\}$ *and* $\{x^l\}$ *generated by the algorithm above have the following properties.*
(*a*) *The sequence* $\{\mu^l\}$ *is bounded and asymptotically maximizing for* (5).
(*b*) *The sequence* $\{x^l\}$ *is bounded and asymptotically minimizing for MCNF.*

Proof: Noting that $\sum_l \varepsilon_L^l < \infty$, result (a) follows from Theorem 2.1 in [59]. Result (b) then follows from (a) and Theorem 4.1 in [60], and the facts that the value of the penalty parameter becomes fixed after a finite number of iterations, and that $\{\varepsilon_L^l\} \to 0$. □

Because of the inexact optimization of the augmented Lagrangian subproblem, we cannot expect the algorithm to find an exact optimal solution to MCNF within a finite number of steps; an optimal solution is however found in the limit, so that we may finitely find near-optimal solutions.

The method for solving the subproblem, the strategies for choosing the initial values of the multipliers and the penalty parameter, other parameter settings, and the termination criteria, are discussed in detail in following sections. In addition, a skeleton of the complete algorithm is illustrated in figure 1 at the end of Section 3.

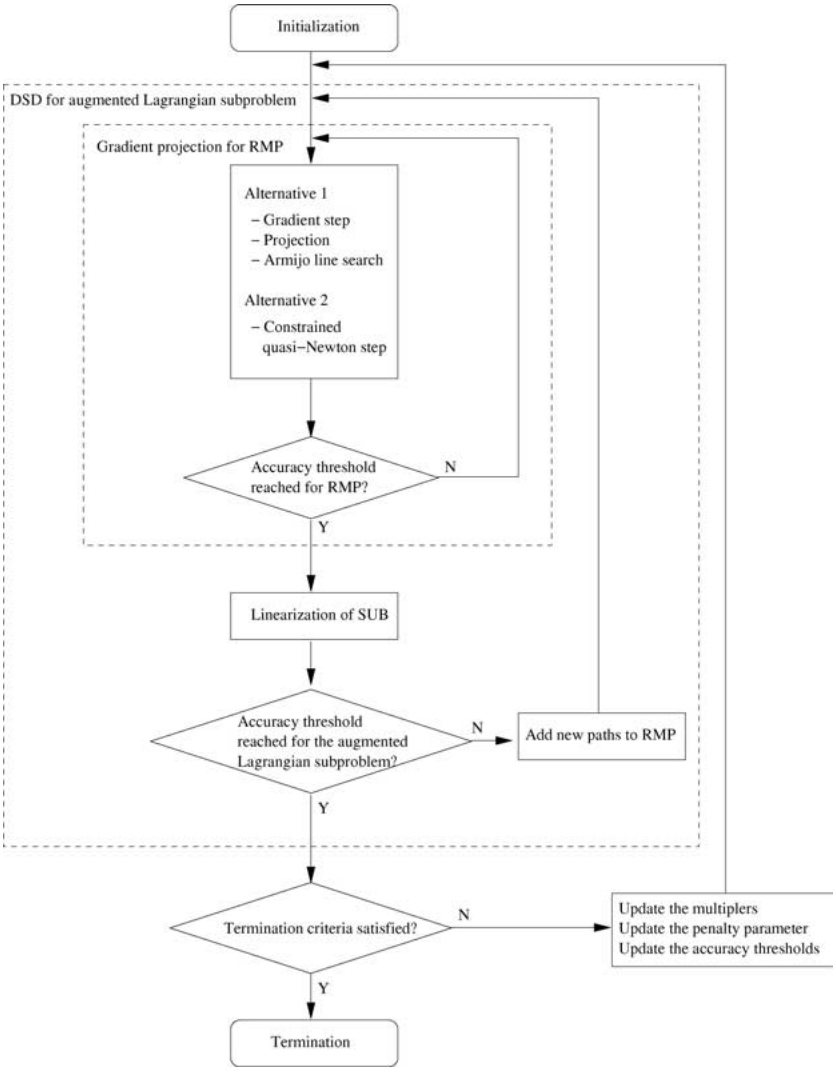

Figure 1. A flow chart of the augmented Lagrangian algorithm.

3.3. Solving the augmented Lagrangian subproblem

The augmented Lagrangian subproblem is a nonlinear multicommodity network flow problem, to which we apply the disaggregate simplicial decomposition (DSD) algorithm [48]. This is a primal method for nonlinear optimization problems over Cartesian product sets. A feature of the DSD algorithm, that makes it particularly suitable in this application, is its reoptimization capability; the solution provided by the DSD algorithm acts naturally as a starting solution when the next subproblem is to be solved.

When simplicial decomposition (SD) [37, 64] is applied to a nonlinear problem over a polytope, extreme points of the polytope are generated through linearizations of the objective function and solutions of the resulting Frank–Wolfe type subproblems. Instead of a simple, one-dimensional search for the next iterate as in the Frank–Wolfe algorithm, a multi-dimensional search is performed by solving a restricted simplicial master problem, which optimizes over the set of all convex combinations of the previously generated extreme points. The SD algorithm alternates between the master problem and the subproblem (the linearized problem).

Consider the set of feasible solutions, X, to SUB. An extreme point of this set is a solution in which the whole demand of each commodity is sent along one elementary path. Observing that X is a Cartesian product with respect to the commodities, that is, $X = X_1 \times X_2 \cdots \times X_{|\mathcal{K}|}$, the DSD algorithm improves SD by exploiting this structure and taking convex combinations of the extreme points in each set X_k separately (instead of taking convex combinations of extreme points in X). A complete SD master problem would, in our case, have $|\mathcal{P}_1| \times |\mathcal{P}_2| \times \cdots \times |\mathcal{P}_{|\mathcal{K}|}|$ columns, because the number of extreme points in X_k equals the number of elementary paths for commodity k. For DSD, the corresponding size is $|\mathcal{P}_1| + |\mathcal{P}_2| + \cdots + |\mathcal{P}_{|\mathcal{K}|}|$. This improvement can be compared to applying Dantzig–Wolfe decomposition to MCNF with disaggregate versus aggregate representations, in which case the former is to prefer [42].

In order to apply DSD to SUB, we assume that subsets of paths, say $\hat{\mathcal{P}}_k \subseteq \mathcal{P}_k$, $k \in \mathcal{K}$, are available. Using the definition (2), a restricted master problem is formulated as follows. (To ease the discussion, the dual iteration index l is omitted in the remainder of this section.)

[RMP]

$$
\begin{aligned}
v_R = \min_x L_c(x, \mu) = \sum_{k \in \mathcal{K}} \sum_{a \in \mathcal{A}} c_{ka} x_{ka} + \frac{1}{2c} \sum_{a \in \mathcal{A}} \left\{ \left[\mu_a + c\left(\frac{x_a}{u_a} - 1 \right) \right]_+^2 - {\mu_a}^2 \right\} \\
\text{s.t.} \quad \sum_{r \in \hat{\mathcal{P}}_k} h_{kr} = d_k \qquad \forall\, k \in \mathcal{K} \\
\sum_{r \in \hat{\mathcal{P}}_k} \delta_{kra} h_{kr} = x_{ka} \qquad \forall\, a \in \mathcal{A},\ \forall\, k \in \mathcal{K} \\
\sum_{k \in \mathcal{K}} x_{ka} = x_a \qquad \forall\, a \in \mathcal{A} \\
h_{kr} \geq 0 \qquad \forall\, r \in \hat{\mathcal{P}}_k,\ \forall\, k \in \mathcal{K}
\end{aligned}
$$

This problem is a restricted version of the augmented Lagrangian subproblem SUB, so that $v_R \geq L_c(\mu)$ holds. Because RMP is not solved exactly in practice, the optimal value v_R is never reached. After solving RMP to a certain accuracy, which gives an upper bound to $L_c(\mu)$, the next step of the algorithm is to perform a linearization of the objective function $L_c(x, \mu)$ with respect to the arc flow variables. The linearized problem is solved over the whole feasible set of SUB, i.e. X, and decomposes into one problem for each $k \in \mathcal{K}$. Because the objective function is convex, solving the linearized problem yields a lower bound to $L_c(\mu)$, together with paths that might be included into the sets $\hat{\mathcal{P}}_k$, $k \in \mathcal{K}$. The DSD algorithm alternates between RMP and the linearized version of SUB, and terminates

when the relative difference between the upper and lower bounds is less than or equal to ε_L. (Note that the upper bound is monotonically improving, but the lower bound is not.)

We now examine the linearization of SUB in more detail. Consider the linearization of the function $L_c(x, \mu)$, with respect to the arc flow variables, at a point $\tilde{x}$ that is feasible (and typically near-optimal) in RMP. The linearized cost for commodity k on arc a is

$$\tilde{c}_{ka} = \frac{\partial L_c(\tilde{x}, \mu)}{\partial x_{ka}} = \begin{cases} c_{ka} + \left[\mu_a + c\left(\dfrac{\tilde{x}_a}{u_a} - 1\right)\right] \Big/ u_a & \text{if } \mu_a + c\left(\dfrac{\tilde{x}_a}{u_a} - 1\right) > 0 \\ c_{ka} & \text{otherwise.} \end{cases} \tag{10}$$

The linearization of SUB, without the constant term, is

$$\min_{x \in X} \sum_{k \in \mathcal{K}} \sum_{a \in \mathcal{A}} \tilde{c}_{ka} x_{ka}. \tag{11}$$

This problem separates into one shortest path problem for each commodity. Let $\bar{x}$ be an optimal solution, and let

$$\underline{L}_c(\tilde{x}, \mu) = L_c(\tilde{x}, \mu) + \sum_{k \in \mathcal{K}} \sum_{a \in \mathcal{A}} \tilde{c}_{ka}(\bar{x}_{ka} - \tilde{x}_{ka}). \tag{12}$$

Because the function $L_c(x, \mu)$ is convex in x, the value of $\underline{L}_c(\tilde{x}, \mu)$ is a lower bound to $L_c(\mu)$; it therefore also provides a lower bound to v^*.

The generation of paths to RMP is based on the following observations. First, we note that

$$\sum_{a \in \mathcal{A}} \tilde{c}_{ka}(\bar{x}_{ka} - \tilde{x}_{ka}) \leq 0 \tag{13}$$

always holds for all $k \in \mathcal{K}$. Second, if equality holds for all $k \in \mathcal{K}$, then $\tilde{x}$ is an optimal solution to SUB. Hence it is natural to expand RMP with paths for which strict inequality holds.

A feature of the DSD algorithm is its reoptimization capability. When RMP is expanded with some new paths, the former near-optimal solution $\tilde{x}$ remains feasible in the expanded RMP, if the values of the new path variables are set to zero. A starting solution for solving the expanded RMP is therefore directly available.

A summary of the DSD algorithm for solving the augmented Lagrangian subproblem SUB is given below. We use $\bar{v}_L$ and $\underline{v}_L$ to denote the best upper and lower bounds generated during the solution process.

1. (*Initialization*). Set $\underline{v}_L = -\infty$.
2. (*Solution of RMP*). Solve RMP to a relative accuracy of at least ε_L. Let $\tilde{x}$ be the solution and $\bar{v}_L = L_c(\tilde{x}, \mu)$.
3. (*Linearization*). Compute the linearized costs according to (10) and solve the shortest path problems (11). Compute $\underline{L}_c(\tilde{x}, \mu)$ according to (12).

4. (*Termination check*). If $\underline{L}_c(\tilde{x}, \mu) > \underline{v}_L$, set $\underline{v}_L = \underline{L}_c(\tilde{x}, \mu)$. If $(\bar{v}_L - \underline{v}_L)/\underline{v}_L \leq \varepsilon_L$, terminate. Otherwise add new paths to RMP and go to Step 2.

Some remarks can be made concerning the above algorithm.

In Step 2, RMP is reoptimized from the previously found solution $\tilde{x}$ (except, of course, the first time RMP is solved). Furthermore, in Step 1, each set $\hat{\mathcal{P}}_k$, $k \in \mathcal{K}$, is carried over from the previous augmented Lagrangian subproblem.

The termination criterion in Step 4 will eventually become fulfilled. The reason for this is that each time the termination criterion is not fulfilled, RMP is enlarged with new paths, and, in addition, the number of paths is finite. Hence either the termination criterion becomes fulfilled before all paths have been generated, or RMP eventually becomes equivalent to the augmented Lagrangian subproblem SUB, in which case the termination criterion must become satisfied. At termination, $(L_c(\tilde{x}, \mu) - L_c(\mu))/L_c(\mu) \leq \varepsilon_L$ holds.

The solution of (11) not only yields a lower bound to $L_c(\mu)$, but also provides paths that can be used to enlarge the sets $\hat{\mathcal{P}}_k$. If in Step 3, $\sum_{a \in \mathcal{A}} \tilde{c}_{ka}(\bar{x}_{ka} - \tilde{x}_{ka}) < 0$ holds for commodity k, then the corresponding path is added to $\hat{\mathcal{P}}_k$ in Step 4. To examine whether this condition holds or not, one may simply compare the cost of the newly generated path to the cost of a minimum cost path among those in $\hat{\mathcal{P}}_k$.

3.4. Solving RMP

The convex nonlinear problem RMP can be solved in various ways. In [48], a scaled reduced gradient algorithm is used. We apply the gradient projection method (e.g. [9]). Let ε_m^l denote the relative accuracy to which RMP is solved. To ensure that RMP is solved to an accuracy of at least ε_L^l (the required relative accuracy for SUB), we set

$$\varepsilon_m^l = \beta_m \varepsilon_L^l \tag{14}$$

with $\beta_m \in (0, 1)$.

To describe the gradient projection method, we consider the convex optimization problem below.

$$\min_{x \in X} f(x) \tag{15}$$

Given a feasible solution, x^p, a fixed step s is taken along a gradient related direction d^p. [The most simple choice is $d^p = -\nabla f(x^p)$.] To reinforce feasibility, the resulting point is then projected onto X, giving

$$\bar{x}^p = [x^p + s d^p]^+, \tag{16}$$

with $[\cdot]^+$ denoting the projection onto X. The next iterate, x^{p+1}, is then a point on the line segment between x^p and $\bar{x}^p$:

$$x^{p+1} = x^p + \alpha^p(\bar{x}^p - x^p) \tag{17}$$

with $\alpha^p \in [0, 1]$ being the step size.

For the method to be practically useful, the projection should have a low computational complexity. In RMP, the feasible solution set is a Cartesian product over the commodities, and for each commodity the feasible set is a simplex of dimension $|\hat{\mathcal{P}}_k|-1$. The projection can therefore be carried out at a low computational cost. (The time complexity for commodity k is linear in $|\hat{\mathcal{P}}_k|$.)

For choosing the step size α^p, we use the Armijo line search method [9], which is based on successive reductions of a tentative step size. Initializing α^p to 1, its value is successively reduced by a factor $\beta_a \in (0, 1)$, that is, it is set to

$$\alpha^p = \beta_a \alpha^p \tag{18}$$

until

$$f(x^p) - f(x^p + \alpha^p(\bar{x}^p - x^p)) \geq -\sigma \alpha^p \nabla f(x^p)^T(\bar{x}^p - x^p) \tag{19}$$

holds for a given $\sigma \in (0, 1)$.

The gradient projection method, as specified above, converges (e.g. [9], Proposition 2.3.1) towards an optimal solution of RMP.

Returning to the problem RMP, we note that the objective can equivalently be considered to be a function of the path flow variables. Moreover, it is convenient to take the step (16) in the space of the path flow variables, where the feasible set is a Cartesian product of simplices. The partial derivative of $L_c(h, \mu)$ with respect to the variable h_{kr} at the current path flow h^p is

$$\frac{\partial L_c(h^p, \mu)}{\partial h_{kr}} = \sum_{a \in \mathcal{A}} \delta_{kra} c_{ka}^p, \tag{20}$$

where the definition of c_{ka}^p is analogous to that of $\tilde{c}_{ka}$ in (10).

Given a gradient related direction d^p in the path flow space and a fixed step s, a new point $\bar{h}^p$ is generated according to (16). The point $\bar{h}^p$ is then converted into a feasible arc flow solution, $\bar{x}^p$. The Armijo line search is performed using the current feasible flow x^p and the direction $\bar{x}^p - x^p$. The reason for performing the line search in the arc flow space is that its dimension, $|A|$, is typically much smaller than the dimension $\sum_{k \in \mathcal{K}} |\hat{\mathcal{P}}_k|$ of the path flow space.

With the choice $d^p = -\nabla_h L_c(h^p, \mu)$, the gradient projection procedure is a first-order method with a relatively slow convergence. We therefore also use an approximate second-order method that utilizes the diagonal elements of the Hessian matrix. The second-order derivative of $L_c(x, \mu)$ with respect to arc flow x_a, at the point x^p, is

$$\frac{\partial^2 L_c(x^p, \mu)}{\partial x_a^2} = \begin{cases} \dfrac{c}{u_a^2} & \text{if } \mu_a + c\left(\dfrac{x_a^p}{u_a} - 1\right) > 0 \\ 0 & \text{otherwise.} \end{cases} \tag{21}$$

Defining

$$q_{kr}^p = \begin{cases} \sum_{a\in\mathcal{A}} \delta_{kra} \dfrac{\partial^2 L_c(x^p,\mu)}{\partial x_a^2} & \text{if } \sum_{a\in\mathcal{A}} \delta_{kra} \dfrac{\partial^2 L_c(x^p,\mu)}{\partial x_a^2} > 0 \\ 1 & \text{otherwise,} \end{cases} \tag{22}$$

the next iterate h^{p+1} is obtained by solving a constrained quasi-Newton problem, which separates into

$$\begin{aligned} \min \quad & \sum_{r\in\hat{\mathcal{P}}_k} \left\{ \frac{\partial L_c(h^p,\mu)}{\partial h_{kr}} \left(h_{kr} - h_{kr}^p\right) + \frac{1}{2s} q_{kr}^p \left(h_{kr} - h_{kr}^p\right)^2 \right\} \\ \text{s.t.} \quad & \sum_{r\in\hat{\mathcal{P}}_k} h_{kr} = d_k \\ & h_{kr} \geq 0 \quad \forall r \in \hat{\mathcal{P}}_k \end{aligned}$$

for each $k \in \mathcal{K}$. The solution to this problem can be characterized as the result obtained from a quasi-Newton step, with step length $s > 0$, followed by a projection onto the feasible set. In our implementation, the above problem is solved by dualizing the demand constraint, and performing a line search on the resulting piecewise quadratic dual objective function, by using a Newton-type method. For each RMP, our implementation switches over to the approximate second-order method after a predefined number of gradient projection iterations.

Both the gradient projection method and the approximate second-order method suit well in the DSD framework, because they can easily be warm-started whenever RMP is extended with new paths.

Gradient projection can alternatively be combined with an Armijo line search along the projection arc [9]. In such an approach, there is no need to generate the point $\bar{x}^p$. The next iterate is instead obtained as $x^{p+1} = [x^p + sd^p]^+$, with the step s determined by the Armijo rule. For RMP, this approach is computationally expensive, because each iteration in this Armijo line search requires a projection operation, and because the line search must be performed in the path flow space instead of the arc flow space.

For a summary of the augmented Lagrangian algorithm, see figure 1.

4. Algorithm details

4.1. Algorithm initialization

To implement the augmented Lagrangian algorithm, the initial values μ^0 and c^0 need to be chosen. We further need to construct the initial sets $\hat{\mathcal{P}}_k$, $k \in \mathcal{K}$, of paths in RMP. We use the following procedure for this initialization. First, a shortest path problem is solved for each commodity (using the arc costs c_{ka}), and an initial arc flow solution, x^0 (typically infeasible), is generated by sending the demands of the commodities along the shortest paths. These

paths are then used to initialize the sets $\hat{\mathcal{P}}_k$, $k \in \mathcal{K}$. Let $\mathcal{A}^0 = \{a \in \mathcal{A} \mid x_a^0 > u_a\}$ and presume that $\mathcal{A}^0$ is non-empty (because otherwise x^0 solves MCNF). For each $a \in \mathcal{A}^0$, the initial multiplier μ_a^0 is chosen as the difference between the cost of the flow x_{ka}^0, $k \in \mathcal{K}$, and the cost of the capacity-feasible flow $(u_a/x_a^0)x_{ka}^0$, $k \in \mathcal{K}$, that is,

$$\mu_a^0 = \sum_{k \in \mathcal{K}} c_{ka} x_{ka}^0 - \sum_{k \in \mathcal{K}} c_{ka} \frac{u_a}{x_a^0} x_{ka}^0. \tag{23}$$

For $a \notin \mathcal{A}^0$, we set $\mu_a^0 = 0$. In the next step, the value of c^0 is assigned such that the Lagrangian and penalty terms have equal values at x^0, that is,

$$\sum_{a \in \mathcal{A}^0} \mu_a^0 \left(\frac{x_a^0}{u_a} - 1 \right) = \frac{c^0}{2} \sum_{a \in \mathcal{A}^0} \left(\frac{x_a^0}{u_a} - 1 \right)^2. \tag{24}$$

To augment the sets $\hat{\mathcal{P}}_k$, $k \in \mathcal{K}$, the function $L_{c^0}(x, \mu)$ is linearized at (x^0, μ^0), and another shortest path problem is solved for each k. If some new path is generated, the initialization procedure terminates; otherwise c^0 is updated according to (8) and the path generation is repeated until $|\hat{\mathcal{P}}_k| > 1$ holds for some commodity.

4.2. Step size adjustment

A crucial parameter in the gradient projection method is the trial step s. We choose to use an iteration-dependent trial step, s^p. Let $L_c(x^p, \mu)$ be the current objective value in RMP (we here again omit the iteration index l), and $\underline{v}_L$ be the best available lower bound generated by (11) and (12). Consider the linearization of the objective function $L_c(x, \mu)$ at x^p. If the direction d^p is used with step size s^p, the value of the linearized function will change with $s^p \nabla_x L_c(x^p, \mu)^T d^p < 0$. As $L_c(x, \mu)$ is convex in x, $L_c(x, \mu) + s^p \nabla_x L_c(x^p, \mu)^T d^p$ is a lower bound to the value of $L_c(x, \mu)$ that is actually achieved by this step. Let $\bar{s}^p \in (0, 1)$ be a required relative reduction in the linearized objective value. Then the step size s^p is chosen such that this reduction is met in the linearization, that is,

$$s^p = \frac{L_c(x^p, \mu) - \underline{v}_L}{\nabla_x L_c(x^p, \mu)^T d^p} \bar{s}^p. \tag{25}$$

Depending of the number of step size reductions in the Armijo line search, the value of the parameter $\bar{s}^p$ is then adjusted adaptively. If the line search ends without any reduction of α^p, it indicates that $\bar{s}^p$ should be increased; otherwise $\bar{s}^p$ should be decreased. Letting m be the number of reductions of α^p, we set

$$\bar{s}^{p+1} = \begin{cases} \beta_u \bar{s}^p & \text{if } m = 0 \\ (\beta_d)^m \bar{s}^p & \text{otherwise,} \end{cases} \tag{26}$$

where $\beta_u > 1$ and $\beta_d \in (0, 1)$.

4.3. Termination criteria

The proposed algorithm does not necessarily generate any feasible solution, and a corresponding upper bound to v^*. A termination criterion that relies on lower and upper bounds to v^* is therefore not possible. Instead, the termination criteria that we use rely on near-feasibility and near-complementary slackness of the solution to the augmented Lagrangian subproblem. It is also required that the current RMP has been solved sufficiently accurately, which in turn determines the accuracy that is required in the augmented Lagrangian subproblem.

Let $\tilde{x}$ be an approximate solution to SUB. The algorithm then terminates if ε_L^l is below a given threshold value ε_q, and if, in addition, the maximal infeasibility and the maximal violation of complementary slackness are bounded by two given thresholds ε_f and ε_c, respectively, that is, if

$$\max_{a\in\mathcal{A}} \frac{\tilde{x}_a - u_a}{u_a} \le \varepsilon_f \quad \text{and} \quad \max_{a\in\mathcal{A}} \left| \frac{\tilde{x}_a - u_a}{u_a} \right| \mu_a \le \varepsilon_c. \tag{27}$$

4.4. Upper bounding: Finding feasible solutions

Our augmented Lagrangian algorithm has asymptotic convergence to primal optimality. In practice it typically terminates with a solution that is not primal feasible, although the infeasibility is usually small. The algorithm can be used as a simple heuristic for generating primal feasible solutions in the following way. Suppose that the arc capacities are artificially reduced to $(1 - \varepsilon_p)u_a$, $a \in \mathcal{A}$, with ε_p being small and positive. If our algorithm is applied to the perturbed problem with the maximal allowed infeasibility $\varepsilon_f = \varepsilon_p/(1 - \varepsilon_p)$, then it will terminate with a solution that does not violate the original mutual capacity constraints, and it will therefore act as a primal heuristic. If the perturbed problem is feasible, our algorithm will produce a primal feasible solution. Note however that there is no guarantee on the quality of this feasible solution. (In addition, running the algorithm on the perturbed problem does not produce a lower bound to the original problem.)

The primary objective when designing the augmented Lagrangian algorithm is to develop a fast and convergent lower-bounding procedure. If, however, the algorithm is run twice, once on the original problem, and once on a perturbed problem, we will obtain both lower and upper bounds to v^*.

4.5. Alternative design and implementation

It should be pointed out that an augmented Lagrangian approach to MCNF can be designed and implemented in alternative ways. Instead of using the quadratic penalty function, one may consider other choices, such as an exponential penalty function [47]. The solution methods for SUB and RMP, as well as the dual search procedure, can also be designed differently.

5. Computational results

5.1. Test problems

Several problem sets have been used for the computational experiments. The first one is based on the so called Canad problems, which originate from [26, 27]. The original Canad problems are capacitated network design instances with fixed charges on arcs, where the MCNF problem arises as the subproblem for a specific network design. In our tests the fixed charges are ignored and the mutual capacities are reduced.

The networks in the second problem set arise in applications of traffic assignment in transportation planning. The motive for using these networks is that the topology of telecommunications networks is often similar to that of transportation networks. The topologies of the networks in this set come from Sioux Falls [52], Barcelona [48], and Winnipeg [57].

The remaining two test sets comprise synthesized networks with specific structures. The third test set includes a number of planar networks with a topology that mimics a typical structure of a telecommunications network. The nodes are randomly generated as points in the plane. The arcs are then created in an increasing order of Euclidean distances, and such that the network becomes planar. The commodities are defined by randomly chosen node pairs. The arc costs are the Euclidean distances between the nodes, while the capacities and the demands are uniformly distributed within certain intervals.

In some telecommunications applications, the network has a logical grid structure (e.g. [36]), and the last problem set is composed by networks of grid type [40]. Other problem data for this set are generated as for the planar networks.

Our main interest is MCNF applications in telecommunications, where the commodities are associated with origin–destination pairs. MCNF instances with other types of commodities (e.g. [12]) are therefore not included in the experiments.

Test problem sizes are summarized in Table 1. Here $|\mathcal{A}^a|$ denotes the percentage of the mutual capacity constraints that are active at an optimal solution.

In Section 5.2 we present the results of our computational experiments with several alternative codes for solving MCNF problems. These codes are optimizing and provide optimal solutions of high accuracy. We then present the computational results of this algorithm in Section 5.3. The main objective when using our augmented Lagrangian algorithm is to quickly find a tight lower bound to v^*, and a near-feasible and near-complementary primal solution; this objective differs from that of the other codes, and we therefore present the results in two separate sections.

All computational experiments are made on a Sun UltraSPARC with a 200 MHz processor and 2 GB physical RAM. The computer is normally configured such that only 1 GB memory is available for any application process. This configuration is used for all the instances in Table 1 and for all the solution methods.

There is a large number of parameters involved in our algorithm, and some preliminary experiments were carried out to find a set of parameter values for which the algorithm performs reasonably well for all the test problems. These parameter values, which are summarized in Table 2, were then used to obtain the bulk of the results. In the following, we use ALM to denote the proposed augmented Lagrangian algorithm for MCNF.

Table 1. Characteristics of the instances.

	$\|\mathcal{N}\|$	$\|\mathcal{A}\|$	$\|\mathcal{K}\|$	$\|\mathcal{A}\| \cdot \|\mathcal{K}\|$	$\|\mathcal{A}^a\|$		$\|\mathcal{N}\|$	$\|\mathcal{A}\|$	$\|\mathcal{K}\|$	$\|\mathcal{A}\| \cdot \|\mathcal{K}\|$	$\|\mathcal{A}^a\|$
Canad						Grid					
can1	20	290	40	11600	6.2	grid1	25	80	50	4000	11.3
can2	20	229	200	45800	11.4	grid2	25	80	100	8000	28.8
can3	20	288	200	57600	22.6	grid3	100	360	50	18000	11.4
can4	30	678	100	67800	6.3	grid4	100	360	100	36000	8.9
can5	30	517	400	206800	10.8	grid5	225	840	100	84000	7.5
can6	30	686	400	277400	9.0	grid6	225	840	200	168000	10.2
Planar						grid7	400	1520	400	608000	7.9
plan30	30	150	92	13800	10.7	grid8	625	2400	500	1.20 M	12.5
plan50	50	250	267	66750	12.4	grid9	625	2400	1000	2.40 M	17.3
plan80	80	440	543	238920	24.3	grid10	625	2400	2000	4.80 M	17.3
plan100	100	532	1085	577200	16.4	grid11	625	2400	4000	9.60 M	15.5
plan150	150	850	2239	1.90 M	18.2	grid12	900	3480	6000	20.88 M	9.2
plan300	300	1680	3584	6.02 M	9.2	grid13	900	3480	12000	41.76 M	12.4
plan500	500	2842	3525	10.02 M	8.4	grid14	1225	4760	16000	76.16 M	9.9
plan800	800	4388	12756	55.97 M	9.0	grid15	1225	4760	32000	152.32 M	10.4
plan1000	1000	5200	20026	104.12 M	9.2						
Traffic											
Sio	24	76	528	40128	27.6						
Bar	1020	2522	7922	19.98 M	0.4						
Win	1052	2836	4344	12.32 M	1.0						

5.2. Computational experiences with other codes

5.2.1. The codes. We have used a general-purpose linear programming code (CPLEX), a specialized primal partitioning code (PPRN), and two codes based on price-directive decomposition.

CPLEX (version 5.0) is one of the most frequently used general LP-solvers. In addition to the primal and dual simplex methods, CPLEX offers a network optimizer. It is observed in [25] that to solve MCNF problems with CPLEX, using the network optimizer followed by dual simplex is the best choice.

The PPRN code (version 1.0, see for example [13, 15]) originates from [16]. It is a primal partitioning code for solving MCNF problems with linear or nonlinear cost functions. In the linear case, it works as a network simplex algorithm specialized for MCNF. The basic idea is to, by performing primal partitioning, maintain and manipulate a working basis that is small compared to the problem size. The size of the working basis depends on the number of active mutual capacity constraints and varies during the solution procedure. Algorithm details can be found in [16].

Table 2. Parameter values in ALM.

Name	Description	Value
ε_L^0	Initial accuracy for the augmented Lagrangian subproblem SUB	0.1
β_L	Reduction factor for ε_L^0 [see (9)]	0.7
β_c	Increase factor for the penalty parameter c^l [see (8)]	1.5
γ	Infeasibility reduction threshold [see (8)]	0.7
β_m	Ratio between accuracy for RMP and ε_L^l [see (14)]	0.5
β_a	Reduction factor for the Armijo step size [see (18)]	0.5
σ	Slope parameter in the Armijo condition (19)	0.2
$\bar{s}^0$	Target for the relative reduction in a linearized objective, used for calculating the initial step in gradient projection (see Section 4.2)	0.5
β_u	Increase factor for $\bar{s}^p$ [see (26)]	2.6
β_d	Reduction factor for $\bar{s}^p$ [see (26)]	0.45
ε_q	Required accuracy for the augmented Lagrangian subproblem SUB (see Section 4.3)	0.02
ε_f	Required accuracy for primal feasibility [see (27)]	0.01
ε_c	Required accuracy for complementary slackness [see (27)]	0.1
ε_p	Scaling factor for the capacities, used to obtain primal feasibility (see Section 4.4)	0.01

Dantzig–Wolfe decomposition is a well-known price-directive decomposition approach to MCNF problems. To evaluate this approach, we made a quite straightforward implementation. As observed in [42], it is preferable to use a disaggregated master problem instead of an aggregated one, and the former strategy is therefore used in our implementation. The master problem is solved to optimality in every iteration (using CPLEX), followed by the solution of a shortest path subproblem for each commodity. All new extreme points generated in the subproblems are added to the master problem.

The bundle method is also based on price-directive decomposition, and has recently been implemented for MCNF models. Details and computational results of this implementation are presented in [23, 25]. Compared to our implementation of Dantzig–Wolfe decomposition, the bundle code is more general, as each commodity may have multiple origins and destinations. It has been shown previously, and is confirmed by our tests, that the bundle method is a competitive approach to MCNF problems.

5.2.2. Results. Computing times (in seconds) of these optimization codes for our test problems are presented in Tables 3 to 6. We use "–" to denote that a problem cannot be solved because of insufficient memory, and "limit" to denote that the computing time is unreasonably long (more than a week). For the instance grid9, CPLEX was interrupted after it had been run for one month. For the instance Win, "error" denotes that the PPRN code reports an optimal objective value which differs considerably from the optimal value found by the other three codes.

Table 3. Computing times of some optimization codes: Canad networks.

Network	CPLEX	PPRN	Dantzig–Wolfe	Bundle
can1	0.37	0.29	0.03	0.10
can2	4.60	7.43	0.11	0.77
can3	13.03	15.07	0.14	2.25
can4	3.34	6.00	0.08	0.80
can5	57.36	98.85	0.30	4.19
can6	29.16	64.07	0.30	5.22

Table 4. Computing times of some optimization codes: traffic networks.

Network	CPLEX	PPRN	Dantzig–Wolfe	Bundle
Sio	74.45	179.40	0.40	2.77
Bar	–	–	79.85	523.86
Win	–	error	995.30	1828.82

Table 5. Computing times of some optimization codes: planar networks.

Network	CPLEX	PPRN	Dantzig–Wolfe	Bundle
plan30	2.71	4.90	0.10	0.26
plan50	75.97	236.66	0.77	2.45
plan80	5933.37	5952.27	9.29	67.67
plan100	30339.58	35013.75	20.70	196.97
plan150	946217.32	519993.41	192.48	1670.38
plan300	–	limit	539.51	3824.19
plan500	–	limit	1914.29	12968.20
plan800	–	–	47626.18	–
plan1000	–	–	142001.47	–

Before comparing the performances of the four codes, we would like to point out that the exact CPU-times should not be paid too much attention, because the codes tested are developed under different circumstances and are of different generality. In addition, each code can be fine-tuned to produce improved CPU-times for a specific set of test data. We are therefore mainly interested in deriving qualitative conclusions.

Both CPLEX and PPRN use LP optimization techniques. PPRN is specially designed for solving MCNF problems, but it does not seem to outperform CPLEX, when the latter uses the network optimizer and the dual simplex method. We observe that although pivot operations are performed very rapidly in PPRN, the algorithm suffers from that MCNF problems are typically highly degenerate. It is however quite evident that PPRN requires less memory than CPLEX. Another observation is that neither CPLEX

Table 6. Computing times of some optimization codes: grid networks.

Network	CPLEX	PPRN	Dantzig–Wolfe	Bundle
grid1	0.23	0.86	0.03	0.03
grid2	2.17	5.69	0.10	0.26
grid3	8.81	23.74	0.23	0.59
grid4	22.54	69.49	0.22	1.06
grid5	258.18	589.75	1.31	5.77
grid6	1433.91	3335.61	3.28	15.63
grid7	34515.28	42092.32	18.12	141.26
grid8	439702.66	325189.66	233.66	707.61
grid9	>2543499.71	limit	919.15	2598.50
grid10	–	limit	1838.89	6438.61
grid11	–	limit	4080.45	19207.20
grid12	–	–	6964.85	45339.80
grid13	–	–	33614.82	–
grid14	–	–	75640.48	–
grid15	–	–	281797.86	–

nor PPRN is capable of solving large scale instances within reasonable computing times.

As can be seen from the results, the two specialized price-directive decomposition methods outperform the general-purpose solver CPLEX and the specialized partitioning code PPRN. For the MCNF problem that we consider, the computing times are lowest for the Dantzig–Wolfe algorithm. Moreover, this algorithm is able to solve all the instances to optimality with the available space of memory. On the other hand, the bundle code is designed for more general MCNF models with multiple origins and destinations for each commodity. For such problems, previous experience [42] indicates that a Dantzig–Wolfe approach might be less attractive. Conversely, if the bundle code were tailored to origin–destination specific commodities, the performance could certainly be improved. (Specific enhancements include the use of a disaggregated representation of the bundle, which will provide better dual search directions, and the solution of shortest path subproblems instead of minimum-cost network flow subproblems [24].) We conjecture therefore that, for a certain type of MCNF problems, tailored implementations of these two methods would achieve similar performances.

In addition to the above observations, we note that the very long solution times required to solve large instances to optimality do motivate the use of methods that quickly can find near-optimal solutions.

The practical performance of the above algorithms can be further analyzed by studying the computing time with respect to the instance size, for example measured in the number of variables in MCNF–A, that is, $S = |\mathcal{A}| \cdot |\mathcal{K}|$. By normalizing the solution times with S^p for various positive values of p, we obtain empirical estimates of the growth rates of the

computational effort. Applying this approach to the set of grid networks, we find that the running time of CPLEX and PPRN are proportional to S^p with $2 < p < 3$, while $1 < p < 2$ holds for the other two methods.

5.3. *Computational results of the augmented lagrangian algorithm*

Computational results of ALM are shown in Tables 7 to 10. Note that ALM is here run twice. In the first run, ALM is applied to the original problem, and it then produces lower bounds that converge to the optimal value, v^*. In the second run, it is applied to a perturbed problem (see Section 4.4), and it then produces a (heuristic) upper bound. The two columns labeled "LBD Time" and "UBD Time" show the computing times (in seconds) of the first respective the second run. Moreover, $\underline{v}$ is the best lower bound found in the first run, and $\bar{v}$ is the upper bound generated by the second run. The columns $(v^* - \underline{v})/v^*$ and $(\bar{v} - v^*)/v^*$ show the quality of the lower and upper bounds, respectively, and the column $(\bar{v} - \underline{v})/\underline{v}$ gives an estimate of the relative accuracy. [Note that $(v^* - \underline{v})/v^* + (\bar{v} - v^*)/v^* \approx (\bar{v} - \underline{v})/\underline{v}$.]

The first run of ALM often terminates with a true relative accuracy of 0.1% or higher. The upper bound found in the second run is often less than 0.3% away from v^*. The lower bounds are typically tighter than the upper bounds; this is due to the fact that the former are convergent to v^*, while the latter are not. Worth mentioning is also that only 10–20 dual iterations (7) are needed for reaching these lower bounds. The solution times for obtaining the results in Tables 7 to 10 are indeed encouraging. For example, the results for the largest instance, with more than 150 million variables in the formulation MCNF–A, are obtained within 30 minutes.

Table 7. Computational results of ALM: Canad networks.

Network	LBD time	UBD time	$\frac{(v^*-\underline{v})}{v^*}$ (%)	$\frac{(\bar{v}-\underline{v})}{\underline{v}}$ (%)	$\frac{(\bar{v}-v^*)}{v^*}$ (%)	Bundle
can1	0.030	0.04	0.53	0.72	0.19	0.070
can2	0.12	0.10	0.067	0.16	0.091	0.78
can3	0.18	0.25	0.10	0.22	0.12	1.99
can4	0.19	0.22	0.047	0.17	0.12	0.76
can5	0.30	0.32	0.069	0.13	0.058	3.50
can6	0.30	0.31	0.11	0.16	0.053	3.25

Table 8. Computational results of ALM: traffic networks.

Network	LBD time	UBD time	$\frac{(v^*-\underline{v})}{v^*}$ (%)	$\frac{(\bar{v}-\underline{v})}{\underline{v}}$ (%)	$\frac{(\bar{v}-v^*)}{v^*}$ (%)	Bundle
Sio	0.18	0.29	0.13	0.43	0.30	1.58
Bar	137.46	131.56	0.014	0.061	0.047	224.47
Win	125.46	158.18	0.0092	0.067	0.057	495.96

Table 9. Computational results of ALM: planar networks.

Network	LBD time	UBD time	$\frac{(v^*-\underline{v})}{v^*}$ (%)	$\frac{(\bar{v}-\underline{v})}{\underline{v}}$ (%)	$\frac{(\bar{v}-v^*)}{v^*}$ (%)	Bundle
plan30	0.090	0.090	0.18	0.20	0.12	0.16
plan50	0.63	0.63	0.11	0.32	0.21	0.85
plan80	6.76	5.49	0.038	0.19	0.15	22.82
plan100	5.56	6.95	0.059	0.21	0.15	99.97
plan150	30.95	30.85	0.074	0.26	0.18	392.59
plan300	52.20	50.89	0.043	0.16	0.11	321.39
plan500	118.30	92.92	0.031	0.17	0.14	699.12
plan800	872.44	699.89	0.044	0.17	0.13	–
plan1000	1417.97	1679.25	0.028	0.18	0.15	–

Table 10. Computational results of ALM: grid networks.

Network	LBD time	UBD time	$\frac{(v^*-\underline{v})}{v^*}$ (%)	$\frac{(\bar{v}-\underline{v})}{\underline{v}}$ (%)	$\frac{(\bar{v}-v^*)}{v^*}$ (%)	Bundle
grid1	0.020	0.020	0.43	0.62	0.18	0.010
grid2	0.040	0.080	0.45	0.60	0.14	0.030
grid3	0.26	0.18	0.13	0.22	0.082	0.23
grid4	0.10	0.13	0.17	0.29	0.12	0.17
grid5	0.70	0.61	0.055	0.12	0.068	1.00
grid6	1.44	0.84	0.075	0.23	0.16	2.79
grid7	3.68	4.42	0.073	0.16	0.086	11.23
grid8	8.01	11.93	0.11	0.20	0.091	33.10
grid9	26.43	25.74	0.12	0.23	0.11	131.27
grid10	61.17	43.24	0.066	0.22	0.15	649.46
grid11	110.31	151.23	0.057	0.14	0.080	1897.87
grid12	102.75	145.82	0.091	0.19	0.098	2448.13
grid13	479.37	468.89	0.080	0.19	0.11	–
grid14	639.91	644.88	0.053	0.16	0.10	–
grid15	1646.42	1188.61	0.032	0.14	0.11	–

A general observation is that small instances can be solved to optimality by the bundle method or Dantzig–Wolfe decomposition with reasonable computing times, while our augmented Lagrangian algorithm is attractive for quickly generating approximate solutions to large scale instances.

The bundle method is a Lagrangian dual approach to MCNF problems, as is ALM. Empirical results suggest that such a method usually proceeds with rapid improvements in its early stages, and most of the solution time is spent for the final convergence. One may therefore wonder how the bundle code compares with ALM, if the former is terminated

before optimality. Such a comparison is enabled by the last column in Tables 7 to 10; these columns display the computing times required for the bundle code to reach the lower bound $\underline{v}$.

We note that ALM is able to solve significantly larger instances than the bundle code could. Moreover, ALM is competitive compared with the bundle code, or even outperforms it, for finding a near-optimal dual solution. We are however not suggesting that ALM will be superior for reaching an exact dual optimum (although the global convergence is ensured), because RMP is solved primarily with a first-order method.

An estimate of the empirical complexity of ALM can be obtained using the procedure described at the end of Section 5.2.2. This leads to the observation that the running time (for the test set of grid networks) is proportional to S^p with $1 < p < 1.2$. However, because the relative accuracy that is obtained when ALM terminates varies by instance, this estimate is less accurate than the previous one.

It is of course interesting to investigate how large instances that ALM can manage. Two network instances with planar and grid topologies are generated for this purpose. The MCNF–A formulations for these two instances contain more than 1 billion variables each (see Table 11). The algorithm was not able to generate solutions to these two instances using 1 GB memory, and the computer was therefore temporarily reconfigured to allow our application program to use 2 GB. The results are presented in Table 12. Both lower and upper bounds to the optimal solution are provided by ALM, with a relative deviation of less than 0.2%. The computing times (several hours) are reasonable, taking the sizes of the instances into account.

The computationally expensive parts in ALM are the solutions of the restricted master problem (RMP) and the linearization of SUB (path generation). Table 13 gives statistics,

Table 11. Two huge network instances.

	$\lvert\mathcal{N}\rvert$	$\lvert\mathcal{A}\rvert$	$\lvert\mathcal{K}\rvert$	$\lvert\mathcal{A}\rvert \cdot \lvert\mathcal{K}\rvert$
planB	2500	12900	81430	1.05 G
gridB	3600	14160	80000	1.10 G

Table 12. Computational results of ALM: huge instances.

Network	LBD time	UBD time	$\frac{(v^*-\underline{v})}{v^*}$ (%)	$\frac{(\bar{v}-\underline{v})}{\underline{v}}$ (%)	$\frac{(\bar{v}-v^*)}{v^*}$ (%)
planB	18633.97	15489.17	–	0.13	–
gridB	13443.73	7994.66	–	0.16	–

Table 13. Portion of time for solving RMP in ALM.

Time portion	≤20%	20–40%	40–60%	60–80%	≥80%
Number of instances	2	2	13	16	2

for the 35 instances, of the portion of the time that is used in ALM (the first run) for solving RMP. It can be seen that a quite good balance is achieved between the computational effort for solving RMP and for path generation. Similar results are obtained for the second run of the algorithm.

6. Conclusions

We have described and evaluated an augmented Lagrangian algorithm for the type of linear multicommodity network flow problem that arises in telecommunications applications. The basic idea is to introduce nonlinear penalties in a Lagrangian relaxation scheme, in order to improve the dual convergence properties, and also to obtain convergence to a primal optimal solution. For solving the augmented Lagrangian subproblem, we use a disaggregate simplicial decomposition algorithm, which exploits the subproblem's structure. Our primary research objective is the construction of an algorithm that is able to quickly produce near-optimal solutions for large scale instances. The described algorithm suits well for this purpose, as demonstrated by the computational results. Good solutions are obtained in reasonable time for very large instances.

A weakness of the proposed augmented Lagrangian algorithm is its lack of finite convergence. Moreover, values of many algorithm parameters need to be chosen; to achieve the best possible performance for a specific type of instances, some calibration of these parameter values is required.

An interesting observation is that the heuristic principle described in Section 4.4 can be extended to produce a sequence of primal feasible solutions which tend to an optimal solution. This can be done by, for example, repeating the heuristic for a diminishing sequence of ε_p. The proper design of such a scheme is a subject to further investigations.

The proposed algorithm has several potential usages. The primary application is to obtain approximate solutions to large scale MCNF instances. Secondly, the algorithm serves nicely as an advanced starting procedure for exact MCNF algorithms, such as Dantzig–Wolfe decomposition; this application is a subject that is worth future studies. Thirdly, the algorithm (and its extensions) can be used to quickly generate lower bounds to integer multicommodity flow problems and fixed-charge multicommodity network design problems; the effective combination of this lower bounding principle with enumeration methods requires further research.

Notes and Acknowledgments

The research in the field of multicommodity network flows is very active, with respect to both theoretical and computational issues, and some developments have therefore occurred since this paper was first written (March, 2000). Among these, we note that J. Castro has developed and published a specialized interior-point algorithm for multicommodity network flows [14]. The essential components of this specialization are an ad hoc preconditioned conjugate gradient solver and a sparse Cholesky factorization. The author reports computational results in which the interior-point algorithm outperforms both PPRN and CPLEX 4.0. For the path

flow formulation of MCNF, an interior point algorithm is presented in [11]; it is shown to require less computing time than the corresponding algorithm for the arc flow formulation. We also note that our choice of the solution method for RMP is (indirectly) supported by the numerical results in a recent paper [58], in which a gradient projection method is shown to be a viable approach to non-linear message routing problems, in comparison to several other solution methods.

A number of new releases of CPLEX have recently become available. According to the announcements from ILOG, the average speed-up factors of the (dual) simplex algorithm of CPLEX 6.0 versus 5.0, CPLEX 6.5 versus 6.0, and CPLEX 7.0 versus 6.5, are 4.5, 2.3, and 1.1, respectively. Experiments with some of the test problems show that, in our application, the speed-up factor of CPLEX 6.5 (the latest version currently available to the authors) versus 5.0 is in the range of 2 to 3. Although this speed-up may affect some of the comparisons made in the paper, it is much too small to change the general conclusions for large scale instances.

We wish to thank Professor Antonio Frangioni for providing the bundle code, and for giving valuable comments and suggestions, which resulted in a number of improvements of the contents and the presentation. We thank Professor Jordi Castro for providing us with the PPRN code and some other useful information. The code for the projection phase of the algorithm was originally written by Dr. Ann-Britt Strömberg. We also wish to thank the two anonymous referees for their valuable comments. Especially we appreciate the second referee's thorough review and many constructive suggestions. The work of the first author is financially supported by the Swedish Research Council (no. 621-2001-2716).

References

1. I. Ali, D. Barnett, K. Farhangian, J. Kennington, B. Patty, B. Shetty, B. McCarl, and P. Wong, "Multicommodity network problems: Applications and computations," IIE Transactions, vol. 16, pp. 127–134, 1984.
2. A.A. Assad, "Multicommodity network flows—A survey," Networks, vol. 8, pp. 37–91, 1978.
3. A.A. Assad, "Solving linear multicommodity flow problems," in Proceedings of IEEE International Conference on Circuits and Computers, N.G. Rabbat (Ed.), 1980, vol. 1, pp. 157–161.
4. C. Barnhart, "Dual-ascent methods for large-scale multicommodity flow problems," Naval Research Logistics, vol. 40, pp. 305–324, 1993.
5. C. Barnhart, C.A. Hane, E.L. Johnson, and G. Sigismondi, "A column generation and partitioning approach for multi-commodity flow problems," Telecommunication Systems, vol. 3, pp. 239–258, 1995.
6. C. Barnhart and Y. Sheffi, "A network-based primal-dual heuristic for the solution of multicommodity network flow problems," Transportation Science, vol. 27, pp. 102–117, 1993.
7. C. Barnhart, C.A. Hane, and P.H. Vance, "Using branch-and-price-and-cut to solve origin-destination integer multicommodity flow problems," Operations Research, vol. 32, pp. 208–220, 1998.
8. D.P. Bertsekas, Constrained Optimization and Lagrange Multiplier Methods. Academic Press: New York, 1982.
9. D.P. Bertsekas, Nonlinear Programming. Athena Scientific: Belmont, MA, 1995.
10. D. Bienstock, Potential Function Methods for Approximately Solving Linear Programming Problems: Theory and Practice. Kluwer Academic Publishers, Boston, MA, 2002.
11. J.F. Bonnans, M. Haddou, A. Lisser, and R. Rébaï, "Interior point methods with decomposition for multicommodity flow problems," Technical report No. 3852, INRIA, Rocquencourt, France, 2000.
12. W.J. Carolan, J.E. Hill, J.L. Kennington, S. Niemi, and S.J. Wichmann, "An empirical evaluation of the KORBX algorithms for military aircraft applications," Operations Research, vol. 38, pp. 240–248, 1990.

13. J. Castro, "PPRN 1.0 user's guide," Technical report, Statistics and Operations Research Department, Universitat Politècnica de Catalunya, Barcelona, Spain, 1994.
14. J. Castro, "A specialized interior-point algorithm for multicommodity network flows," SIAM Journal on Optimization, vol. 10, pp. 852–877, 2000.
15. J. Castro and N. Nabona, "Computational tests of a linear multicommodity network flow code with linear side constraints through primal partitioning," Technical report, Statistics and Operations Research Department, Universitat Politècnica de Catalunya, Barcelona, Spain, 1994.
16. J. Castro and N. Nabona, "An implementation of linear and nonlinear multicommodity network flows," European Journal of Operational Research, vol. 92, pp. 37–53, 1996.
17. P. Chardaire and A. Lisser, "Simplex and interior point specialized algorithms for solving non-oriented multicommodity flow problems," Operations Research, vol. 50, pp. 260–276, 2002.
18. H. Chen and C.G. DeWald, "A generalized chain labelling algorithm for solving multicommodity flow problems," Computers and Operations Research, vol. 1, pp. 437–465, 1974.
19. H. Crowder, "Computational improvements for subgradient optimization," in Symposia Mathematica, Academic Press: London, 1976, vol. XIX, pp. 357–372.
20. J.M. Farvolden, W.B. Powell, and I.J. Lustig, "A primal partitioning solution for the arc-chain formulation of a multicommodity network flow problem," Operations Research, vol. 41, pp. 669–693, 1993.
21. L.K. Fleischer, "Approximating fractional multicommodity flow independent of the number of commodities," SIAM Journal on Discrete Mathematices, vol. 3, pp. 505–520, 2000.
22. M. Folie and J. Tiffin, "Solution of a multi-product manufacturing and distribution problem," Management Science, vol. 23, pp. 286–296, 1976.
23. A. Frangioni, "Dual-ascent methods and multicommodity flow problems," PhD thesis, Dipartimento di Informatica, Università di Pisa, Italy, 1997.
24. A. Frangioni, private communications, 1999.
25. A. Frangioni and G. Gallo, "A bundle type dual-ascent approach to linear multicommodity min-cost flow problems," Informs Journal on Computing, vol. 11, pp. 370–393, 1999.
26. B. Gendron and T.G. Crainic, "Bounding procedures for multicommodity capacitated network design problems," Technical report, Centre de recherche sur les transports, Université de Montréal, Canada, 1995.
27. B. Gendron, T.G. Crainic, and A. Frangioni, "Multicommodity capacitated network design," in Telecommunications Network Planning, B. Sansó and P. Sariano (Eds.), Kluwer Academic Publishers, 1999, chap. 1, pp. 1–19.
28. A.M. Geoffrion and G.W. Graves, "Multicommodity distribution system design by Benders decomposition," Management Science, vol. 20, pp. 822–844, 1974.
29. A.V. Goldberg, J.D. Oldham, S. Plotkin, and C. Stein, "An implementation of a combinatorial approximation algorithm for minimum-cost multicommodity flow," in Integer Programming and Combinatorial Optimization, Proceedings of the 6th International IPCO Conference, R.E. Bixby, E.A. Boyd, and R.Z. Ríos-Mercado (Eds.), Houston, Springer, 1998, pp. 338–352.
30. M.D. Grigoriadis and L.G. Khachiyan, "En exponential-function reduction method for block-angular convex programs," Networks, vol. 26, pp. 59–68, 1995.
31. M.D. Grigoriadis and W.W. White, "A partitioning algorithm for the multicommodity network flow problem," Mathematical Programming, vol. 3, pp. 157–177, 1972.
32. M.D. Grigoriadis and W.W. White, "Computational experience with a multicommodity network flow algorithm," in Optimization Methods for Resource Allocation, R. Cottle and J. Krarup (Eds.), The English Universities Press Ltd: London, 1974, pp. 205–227.
33. R.C. Grinold, "A multicommodity max-flow algorithm," Operations Research, vol. 16, pp. 1234–1237, 1968.
34. J.K. Hartman and L.S. Lasdon, "A generalized upper bounding algorithm for multicommodity network flow problems," Networks, vol. 1, pp. 333–354, 1972.
35. M. Held, P. Wolfe, and H.P. Crowder, "Validation of subgradient optimization", Mathematical Programming, vol. 6, pp. 62–88, 1974.
36. M. Henningson, K. Holmberg, and M. Näsberg, "A capacitated bus grid network design problem," in Proceedings of 8th International Conference on Telecommunication Systems: Modeling and Analysis, Nashville, USA, 2000, pp. 98–113.

37. C.A. Holloway, "An extension of the Frank and Wolfe method of feasible directions," Mathematical Programming, vol. 6, pp. 14–27, 1974.
38. K. Holmberg, "Lagrangean heuristics for linear cost multicommodity network flow problem," Working paper LiTH-MAT/OPT-WP-1995-01, Division of Optimization, Department of Mathematics, Linköping Institute of Technology, Sweden, 1995.
39. K. Holmberg, M. Joborn, and J.T. Lundgren, "Improved empty freight car distribution," Transportation Science, vol. 32, pp. 163–173, 1998.
40. K. Holmberg and D. Yuan, "A multicommodity network-flow problem with side constraints on paths solved by column generation," Informs Journal on Computing, vol. 15, pp. 42–57, 2003.
41. K. Holmberg and D. Yuan, "A Lagrangean heuristic based branch-and-bound approach for the capacitated network design problem," Operations Research, vol. 48, pp. 461–481, 2000.
42. K.L. Jones, I.J. Lustig, J.M. Farvolden, and W.B. Powell, "Multicommodity network flows: The impact of formulation on decomposition," Mathematical Programming, vol. 62, pp. 95–117, 1993.
43. A. Kamath, O. Palmon, and S. Plotkin, "Fast approximation algorithm for minimum cost multicommodity flow," in Proceedings of the Sixth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA), San Francisco, 1995, pp. 493–501.
44. J.L. Kennington, "Solving multicommodity transportation problems using a primal partitioning simplex technique," Naval Research Logistics Quarterly, vol. 24, pp. 309–325, 1977.
45. J.L. Kennington, "A survey of linear cost multicommodity network flows," Operations Research, vol. 26, pp. 209–236, 1978.
46. J.L. Kennington and M. Shalaby, "An effective subgradient procedure for minimal cost multicommodity flow problems," Management Science, vol. 23, pp. 994–1004, 1977.
47. B.W. Kort and D.P. Bertsekas, "Combined primal-dual and penalty methods for convex programming," SIAM Journal on Control and Optimization, vol. 14, pp. 268–294, 1976.
48. T. Larsson and M. Patriksson, "Simplicial decomposition with disaggregated representation for the traffic assignment problem," Transportation Science, vol. 26, pp. 4–17, 1992.
49. T. Larsson and M. Patriksson, "An augmented Lagrangean dual algorithm for link capacity side constrained traffic assignment problems," Transportation Research B, vol. 29, pp. 433–455, 1995.
50. T. Larsson, M. Patriksson, and A.-B. Strömberg "Ergodic, primal convergence in dual subgradient schemes for convex programming," Mathematical Programming, vol. 86, pp. 283–312, 1999.
51. T. Leighton, F. Makedon, S. Plotkin, C. Stein, É. Tardos, and S. Tragoudas, "Fast approximation algorithms for multicommodity flow problems," Journal of Computer and System Sciences, vol. 50, pp. 228–243, 1995.
52. L.J. LeBlanc, E.K. Morlok, and W.P. Pierskalla, "An efficient approach to solving the road network equilibrium traffic assignment problem," Transportation Science, vol. 19, pp. 445–462, 1975.
53. S.F. Maier, "A compact inverse scheme applied to a multicommodity network with resource constraints," in Optimization Methods for Resource Allocation, R. Cottle and J. Krarup (Eds.), pp. 179–203, The English Universities Press Ltd., London, 1974.
54. R.D. McBride and J.W. Mamer, "Solving multicommodity flow problems with a primal embedded network simplex algorithm," INFORMS Journal on Computing, vol. 9, pp. 154–163, 1997.
55. R.D. McBride and J.W. Mamer, "A decomposition-based pricing procedure for large-scale linear program: An application to the linear multicommodity flow problem," Management Science, vol. 46, pp. 693–709, 2000.
56. C.J. McCallum, Jr., "A generalized upper bounding approach to a communications network planning problem," Networks, vol. 7, pp. 1–23, 1977.
57. S. Nguyen, "A unified approach to equilibrium methods for traffic assignment," in Traffic Equilibrium Methods, Proceedings of the International Symposium in Montréal, 1976, M. A. Florian (Ed.), Springer, 1976, pp. 148–182.
58. A. Ouorou, P. Mahey, and J.-Ph. Vial, "A survey of algorithms for convex multicommodity flow problems," Management Science, vol. 46, pp. 126–147, 2000.
59. R.T. Rockafellar, "The multiplier method of Hestenes and Powell applied to convex programming," Journal of Optimization Theory and Applications, vol. 12, pp. 555–562, 1973.
60. R.T. Rockafellar, "A dual approach to solving nonlinear programming problems by unconstrained optimization," Mathematical Programming, vol. 5, pp. 354–373, 1973.

61. R.R. Schneur and J.B. Orlin, "A scaling algorithm for multicommodity flow problems," Operations Research, vol. 46, pp. 231–246, 1998.
62. B. Shetty and R. Muthukrishnan, "A parallel projection for the multicommodity network model," Journal of the Operational Research Society, vol. 41, pp. 837–842, 1990.
63. J.A. Tomlin, "Minimum-cost multicommodity network flows," Operations Research, vol. 14, pp. 45–51, 1966.
64. B. Von Hohenbalken, "Simplicial decomposition in nonlinear programming algorithms," Mathematical Programming, vol. 13, pp. 49–68, 1977.