CHAPTER II

SOLUTION METHODS FOR MULTICOMMODITY NETWORK FLOW PROBLEMS

In this chapter, we try to summarize most of the solution techniques that have appeared in the literature, especially those which appeared after 1980.

Due to the special block-angular structure of MCNF problems, many special solution methods have been suggested in the literature. *Basis partitioining methods* partition the basis in a way such that computing the inverse of the basis is faster. *Resource-directive methods* seek optimal capacity allocations for commodities. *Price-directive methods* try to find the optimal penalty prices (dual variables) for violations of the bundle constraints. *Primal-dual methods* raise the dual objective while maintaining complementary slackness conditions and will be discussed in more detail in Chapter 6. In the last two decades, many new methods such as *approximation methods*, *interior-point methods* and their parallelization have been proposed. Although the convex or integral MCNF problems are not the main interests of this dissertation, we still summarize recent progress in these topics.

2.1 Basis partitioning methods

The underlying idea of basis partitioining methods is to partition the basis so that the special network structure can be maintained and exploited as much as possible to make the inversion of the basis more efficient. Hartman and Lasdon [167] propose a Generalized Upper Bounding (GUB) technique which is a specialized simplex method whose only nongraphic operations involve the inversion of a working basis with dimension equal to the number of currently saturated arcs. Other basis partitioning methods based on GUB techniques have also been suggested by McCallum [238] and Maier [228].

Grigoriadis and White [156, 155] observe that, in practice, the number of active bundle constraints in the optimal solution are considerably smaller than the number of capacitated

arcs. Therefore they partition the constraints into two groups: current and secondary constraints. Using the dual simplex method with Rosen's partition technique [273], they iteratively solve sequences of LPs that use only updated current constraints. These LPs do not need to be solved to optimality. The algorithm only requires a basis change when dual feasibility is maintained and the objective value decreases.

Kennington [200] implements the primal partitioning methods of Saigal [277] and Hartman and Lasdon [167]. His implementation of basis partitioning methods are computationally inferior to resource-directive subgradient methods.

EMNET, developed by McBride [233] and based on the factorization methods (variants of basis partition methods) of Graves and McBride [154], is designed to quickly solve LP problems with network structure and network problems with side constraints. Combined with other techniques such as resource-directive heuristics and price-directive column generation, they conclude that basis partitioning methods are computationally efficient. In particular, McBride and Mammer [236] use a capacity allocation heuristic to produce a hot-start basis for EMNET which shortens the computational time. McBride [235] uses a reource-directive decomposition heuristic to control the size of the working inverse and dramatically reduce the solution time for MCNF problems with many side constraints. Mamer and McBride [229, 237] also apply column generation techniques of DW decomposition to solve message routing problems and find a dramatic reduction in the number of pivots needed. The combination of EMNET and column generation shows a larger reduction in computation time than direct application of EMNET. A similar specialized simplex algorithm has also been proposed by Detlefsen and Wallace [92]. Their paper addresses more details of the network interpretation of basis inversion. Independent of the number of commodities, their working basis has dimesion at most equal to the number of arcs in the network, which seems to be suitable for telecommunication problems that often have a large number of commodities.

Castro and Nabona [70] implement a MCNF code named PPRN to solve min-cost MCNF problems which have a linear or nonlinear objective function and additional linear side constraints. PPRN is based on a primal partitioning method using Murtagh and Saunders'

[246] strategy of dividing the variables into three sets: basic, nonbasic and superbasic.

Farvolden et al. [109] partition the basis of the master problem using the arc-path form in DW decomposition. Their techniques isolate a very sparse and near-triangular working basis of much smaller dimension, and identify nine possible types of simplex pivots that can be done by additive and network operations to greatly improve the efficiency. Such techniques are later used to solve a MCNF problem with jump constraints (i.e., commodities can not be shipped along more than a fixed number of arcs) [232] and further refined by Hadjiat et al. [163] so that the dimension of the working basis is at most equal to the number of arcs in the network (almost half the size of the working matrix of [109]).

2.2 Resource-directive methods

Suppose on each arc (i,j) we have assigned capacity r_{ij}^k for each commodity k so that $\sum_{k\in K} r_{ij}^k \le u_{ij}$ is satisfied. Then the original problem is equivalent to the following *resource allocation problem* (RAP) that has a simple constraint structure but complex objective function:

$$\begin{aligned} \min \sum_{k\in K} z^k(r^k) &= z(r) \qquad\qquad (\text{RAP}) \\ s.t. \quad \sum_{k\in K} r_{ij}^k &\le u_{ij} \quad \forall (i,j)\in A \\ 0 \le r_{ij}^k &\le u_{ij} \quad \forall (i,j)\in A,\ \forall k\in K \end{aligned}$$

For each commodity k, $z^k(r^k)$ can be obtained by solving the following single commodity min-cost network flow subproblem.

$$\begin{aligned} \min \sum_{(i,j)\in A} c_{ij}^k x_{ij}^k &= z^k(r^k) \\ s.t. \sum_{(i,j)\in A} x_{ij}^k - \sum_{(j,i)\in A} x_{ji}^k &= b_i^k \quad \forall i\in N \\ 0 \le x_{ij}^k &\le r_{ij}^k \quad \forall (i,j)\in A \end{aligned}$$

It can be shown that the objective function $z(r)$ is piecewise linear on the feasible set of the capacity allocation vector r. There are several methods in the literature to solve

the RAP such as *tangential approximation* [131, 132, 201], *feasible directions* [201], and the *subgradient method* [131, 161, 153, 168, 199]. Shetty and Muthukrishnan [289] give a parallel projection algorithm which parallelizes the procedure of projecting new resource allocations in the resource-directive methods.

2.3 Price-directive methods

Price-directive methods are based on the idea that by associating the bundle constraints with "correct" penalty functions (dual prices, or Lagrange multipliers), a hard MCNF problem can be decomposed into k easy SCNF problems.

2.3.1 Lagrange relaxation (LR)

Lagrange relaxation dualizes the bundle constraints using a Lagrangian multiplier $\pi \geq 0$ so that the remaining problem can be decomposed into k smaller min-cost network flow problems. In particular,

$$\begin{aligned} \min & \sum_{k \in K} c^k x^k + \sum_{a \in A} \pi_a (\sum_{k \in K} x_a^k - u_a) = L(\pi) \\ s.t. \quad & \widetilde{N} x^k = b^k, \, \forall k \in K \\ & x \geq 0 \end{aligned}$$

where $\widetilde{N}$ is the node-arc incidence matrix. The Lagrangian dual problem seeks an optimal π^* for $L^* = \max_{\pi \geq 0} L(\pi)$. This is a *nondifferentiable optimization problem* (NDO) having the format $\max\{\varphi(y) : y \in Y\}$ where φ is a concave nondifferentiable function and Y is a convex nonempty subset of $R_+^{|A|}$.

Subgradient methods are common techniques for determining the Lagrange multipliers. They are easy to implement but have slow convergence rates. They usually do not have a good stopping criterion, and in practice usually terminate when a predefined number of iterations or nonimproving steps is reached. Papers regarding subgradient methods have been listed in Section 2.2. Chapter 16 and 17 of [3] also present illustrative examples.

Bundle methods are designed to solve general NDO problems and thus are suitable for solving the Lagrangian dual problems. In particular, let $g(y')$ denote a supergradient of φ

at y', that is, $\varphi(y) \leq \varphi(y') + g(y - y')$ for all y in Y. Define the *bundle* to be a set β that contains all $\varphi(y_i)$ and $g(y_i)$ generated in previous iterations. The generic bundle method starts with an initial y' and β and searches for a tentative ascent direction d. It tentatively updates $y'' = y' + \theta d$. If the improvement $\varphi(y'') - \varphi(y')$ is large enough, the solution y' will move to y''. Otherwise, it adds $\varphi(y'')$ and $g(y'')$ to the bundle β so that next iteration may give a better tentative ascent direction.

In general, bundle methods converge faster and are more robust than subgradient methods. Frangioni's Ph.D. dissertation [118] is a rich resource for the bundle method and its application to solving MCNF problems. Gendron et al. [130] use the bundle method to solve a multicommodity capacitated fixed charge network design problem. Cappanera and Frangioni [59] give its parallelization together with discussion on other parallel price-directive approaches.

Both the subgradient methods and bundle methods are dual based techniques, in which usually extra effort has to be made to obtain a primal feasible solution. Next we will introduce methods that improve the primal feasible solution and compute the Lagrangian multipliers by solving a LP in each iteration.

2.3.2 Dantzig-Wolfe decomposition (DW)

As previously mentioned in Section 1.4.3, here we illustrate more details about the DW procedures using the arc-path form. The primal path formulation is as follows:

$$\min \sum_{k \in K} \sum_{p \in P^k} PC_p^c f_p = Z_P^*(f, s) \qquad \text{(P\_PATH)}$$

$$s.t. \qquad \sum_{p \in P^k} f_p = 1 \quad \forall k \in K \qquad (2.1)$$

$$\sum_{k \in K} \sum_{p \in P^k} (B^k \delta_a^p) f_p \leq u_a \quad \forall a \in A \qquad (2.2)$$

$$f_p \geq 0 \quad \forall p \in P^k, \forall k \in K$$

whose dual is

$$\max \sum_{k\in K} \sigma_k + \sum_{a\in A} u_a(-\pi_a) = Z_D^*(\pi,\sigma) \tag{D\_PATH}$$

$$s.t.\ \ \sigma_k + \sum_{a\in A} B^k \delta_a^p(-\pi_a) \le PC_p^c \ \ \forall p \in P^k, \forall k \in K \tag{2.3}$$

$$\pi_a \ge 0 \ \ \forall a \in A$$

$$\sigma_k : \text{free} \ \ \forall k \in K$$

where σ_k are the dual variables for the convexity constraint (2.1) and $-\pi_a$ are the dual variables for the bundle constraints (2.2).

The complementary slackness (CS) conditions are:

$$(-\pi_a)(\sum_{k\in K}\sum_{p\in P^K} (B^k\delta_a^p) f_p - u_a) = 0 \ \ \forall a \in A \tag{CS.1}$$

$$\sigma_k(\sum_{p\in P^K} f_p - 1) = 0 \ \ \forall k \in K \tag{CS.2}$$

$$f_p(PC_p^{c+\pi} - \sigma_k) = 0 \ \ \forall p \in P^k, \ \ \forall k \in K \tag{CS.3}$$

where $PC_p^{c+\pi} := \sum_{a\in A} B^k \delta_a^p (c_a + \pi_a)$ and c_a is the original cost of arc a. The reduced cost of path p is $PC_p^c - \sum_{a\in A} B^k\delta_a^p(-\pi_a) - \sigma_k = \sum_{a\in A} B^k\delta_a^p(c_a+\pi_a) - \sigma_k = PC_p^{c+\pi} - \sigma_k$.

Suppose a feasible but nonoptimal primal solution (f, s) is known. We construct the RMP which contains the columns associated with positive f and s, solve the RMP to optimality, and obtain its optimal dual solution $(\sigma^*, -\pi^*)$. This optimal dual solution can be used to identify profitable columns (i.e. columns with reduced cost $PC_p^{c+\pi^*} - \sigma_k^* < 0$) and add them to construct a new RMP. The procedure for generating new columns is equivalent to solving k subproblems. Each subproblem corresponds to a shortest path problem from s^k to t^k for a single commodity k. In particular, using $(c_{ij} + \pi_{ij}^*)$ as the new cost for each arc (i, j), if the length of the shortest path p^{k*} for commodity k, $PC_{p^{k*}}^{c+\pi^*} = \sum_{(i,j)\in A} B^k \delta_{ij}^{p^{k*}} (c_{ij} + \pi_{ij}^*)$, is shorter than σ_k^*, then we add the column corresponding to path p^{k*} to the RMP. The algorithm iteratively solves the RMP and generates new columns until no more columns have negative reduced cost (which guarantees optimality). During the procedure, primal feasibility and complementary slackness are maintained.

The DW algorithm stated above can also be viewed as a procedure to identify the optimal dual variables $(\sigma^*, -\pi^*)$ by solving sequences of smaller LP problems (RMP). Compared to LR, DW spends more time in solving the RMP, but obtains better dual variables. In general, DW requires fewer iterations than LR to achieve optimality. Another advantage of DW is that it always produces primal feasible solutions which can be used as upper bounds in minimization, while its dual objective can be used as a lower bound. LR, on the other hand, usually only guarantees an optimal dual solution and may require extra effort to obtain an optimal primal solution.

Unlike the conventional dual-based subgradient methods, the *volume algorithm* of Barahona and Anbil [24] is an extension of the subgradient methods. It produces approximate primal solutions in DW procedures. The method is similar to the subgradient method and the bundle method introduced in Section 2.3.1, and has the following steps:

- **Step 0**: Given a feasible dual solution $-\overline{\pi}$ to the bundle constraint (2.2), we solve $|K|$ subproblems. Each subproblem has the following form:

$$\min \sum_{(i,j)\in A} (c_{ij}^k + \overline{\pi}_{ij}) x_{ij}^k = \overline{Z}^k \qquad \text{(Vol}(k)\text{)}$$

$$s.t. \sum_{(i,j)\in A} x_{ij}^k - \sum_{(j,i)\in A} x_{ji}^k = b_i^k \quad \forall i \in N$$

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

  Suppose (Vol(k)) has optimal solution $\overline{x}^k$ and optimal objective value $\overline{Z}^k$.
  Set $t = 1$, $\overline{x} = [\overline{x}^1 \ldots \overline{x}^{|K|}]^T$ and $\overline{Z} = \sum_{k=1}^{|K|} \overline{Z}^k - \sum_{(i,j)\in A} u_{ij}\overline{\pi}_{ij}$.

- **Step 1**: Compute $v_{ij}^t = u_{ij} - \sum_{k\in K} \overline{x}_{ij}^k$, and $\pi_{ij}^t = \max\{\overline{\pi}_{ij} + \theta v_{ij}^t, 0\}$ for each arc (i,j) where the step length $\theta = \overline{f}\frac{UB-\overline{Z}}{\|v^t\|^2}$, $\overline{f}$ is a parameter between 0 and 2, and UB is the upper bound on the objective for P_PATH which can be computed by any primal feasible solution.

  For each commodity k, we solve (Vol(k)) using the new π_{ij}^t instead of $\overline{\pi}_{ij}$ in the objective function, and obtain its optimal solution x^{kt} and optimal objective value

Z^{kt}

Set $x^t = [x^{1t} \dots x^{|K|t}]^T$ and $Z^t = \sum_{k=1}^{|K|} Z^{kt} - \sum_{(i,j)\in A} u_{ij}\pi_{ij}^t$.

Update $\overline{x} = \eta x^t + (1-\eta)\overline{x}$ where η is a parameter between 0 and 1.

- **Step 2**: if $Z^t > \overline{Z}$ then update $\overline{\pi}_{ij} = \pi_{ij}^t$ for each arc (i,j) and $\overline{Z} = Z^t$.

  Let $t = t+1$, and go to Step 1.

The algorithm terminates when either $\left\|v^t\right\|$ becomes too small or $\left|\sum_{(i,j)\in A} u_{ij}\overline{\pi}_{ij}\right|$ is smaller than some threshold value.

Suppose $Z^t \le \overline{Z}$ for all iterations $t_c \le t \le t_d$. In these iterations, the volume algorithm uses the new $\overline{x}_{ij}^k$ but the old $\overline{\pi}_{ij}$ and $\overline{Z}$ to compute the new subgradients v_{ij}^t. In such case, $\overline{x} = (1-\eta)^{t_d-t_c}x^{t_c} + (1-\eta)^{t_d-t_c-1}\eta x^{t_c+1} + \dots + (1-\eta)\eta x^{t_d-t_c-1} + \eta x^{t_d-t_c}$. That is, $\overline{x}$ is a convex combination of $\{x^{t_c}, \dots, x^{t_d}\}$ which is an approximation to the optimal arc flow solution.

The basic idea is, given $(\overline{Z}, -\overline{\pi})$ where $-\overline{\pi}$ is a feasible dual variable to the bundle constraint and $\overline{Z}$ is computed using $-\overline{\pi}$, the volume algorithm moves in a neighborhood of $(\overline{Z}, -\overline{\pi})$ to estimate the volumes below the faces that are active in the neighborhood; thus better primal solutions are estimated at each iteration. On the other hand, using the conventional subgradient methods, the subgradient v is usually determined by only one active face, and no primal solution can be computed.

The volume algorithm has been tested with success in solving some hard combinatorial problems such as crew scheduling problems [25], large scale facility location problems [26, 27], and Steiner tree problems [23]. In [23], the problem is formulated as a non-simultaneous MCNF problem. It is believed that the volume algorithm is advantageous for solving set partitioning-type problems including the MCNF problems. It will be interesting to learn how this new algorithm performs in solving general MCNF problems. To the best of our knowledge, no such work has been done yet.

2.3.3 Key variable decomposition

Based on a primal partitioning method of Rosen [273], Barnhart et al. [36] give a column generation algorithm which is especially suitable for problems with many commodities (OD

pairs). In particular, they select a candidate path called a *key path*, $key(k)$, for each commodity k. After performing some column operations to eliminate $key(k)$ for each k, they obtain the following CYCLE form:

$$\min \sum_{k\in K}\sum_{p\in P^k} CC_p^c f_p + \sum_{a\in A} M\alpha_a = Z_C^*(f,\alpha) \qquad \text{(CYCLE)}$$

$$s.t. \quad \sum_{p\in P^k} f_p = 1 \quad \forall k \in K \qquad (2.4)$$

$$\sum_{k\in K}\sum_{p\in P^k} B^k(\delta_a^p - \delta_a^{key(k)}) f_p - \alpha_a \le u_a - \sum_{k\in K}(B^k \delta_a^{key(k)}) \ \forall a \in A \qquad (2.5)$$

$$f_p \ge 0 \quad \forall p \in P^k, \forall k \in K$$

$$\alpha_a \ge 0 \quad \forall a \in A$$

where α_a is an artificial variable for each arc a with large cost M, and $CC_p^c = PC_p^c - PC_{key(k)}^c$ represents the cost of several disjoint cycles obtained by the symmetric difference of path p and $key(k)$. This CYCLE formulation is essentially the same as the P_PATH formulation in Dantzig-Wolfe decomposition. The artificial variables with large costs here are used to get an easy initial primal feasible solution for the algorithm.

When the number of commodities (OD pairs) is huge, both CYCLE and P_PATH will have many constraints, which makes the RMP more difficult. To overcome this computational burden, Barnhart et al. exploit Rosen's key variable concept which relaxes the nonnegativity constraints for each key path (i.e., it allows $f_{key(k)}$ to be negative). The RMP can be solved by iteratively solving these easier problems RELAX(i), each of which contains only (1) all of the bundle constraints and (2) the nonnegativity constraints for all variables except the key variables.

$$\min \sum_{k\in K}\sum_{p\in P^k} CC_p^{c,i} f_p^i + \sum_{a\in A} M\alpha_a^i = Z_{CR(i)}^*(f,\alpha) \qquad \text{(RELAX}(i)\text{)}$$

$$s.t. \sum_{k\in K}\sum_{p\in P^k} B^k(\delta_a^p - \delta_a^{key(k,i)}) f_p^i - \alpha_a^i \le u_a - \sum_{k\in K}(B^k \delta_a^{key(k,i)}) \ \forall a \in A \qquad (2.6)$$

$$f_p^i \ge 0 \quad \forall p \in P^k \setminus key(k,i), \forall k \in K \ ; \quad f_{key(k,i)} : \text{free} \quad \forall k \in K$$

$$\alpha_a^i \ge 0 \quad \forall a \in A$$

After solving RELAX(i), the algorithm will check the sign of key variables by calculating $f^{i*}_{key(k,i)} = 1 - \sum_{p \in P^k \setminus key(k,i)} f^{i*}_p$. For those key variables with negative signs, the algorithm will perform a *key variable change* operation which replaces the current key variable with a new one. Among all the positive path variables, the one with the largest value is usually chosen since intuitively that path is more likely to have positive flow in the optimal solution. This algorithm maintains dual feasibility and complementary slackness while trying to achieve primal feasibility (which will be attained when all key variables become nonnegative).

Like the subproblems of DW, the subproblems of RELAX(i) are shortest path problems for each commodity k. Suppose in some iteration we solve the RMP of RELAX(i) and obtain the optimal dual solution associated with the bundle constraint (2.6), denoted by $-\pi^*$. A shortest path p^{k*} is generated if its reduced cost $PC^{c+\pi^*,i}_{p^{k*}} - PC^{c+\pi^*,i}_{key(k,i)} < 0$. Barnhart et al. [34, 35] further use a *simple cycle heuristic* to generate more good columns to shorten the overall computational time. In particular, for commodity k, the symmetric difference of shortest path p^{k*} and key path $key(k,i)$ forms several simple cycles. Intuitively, these simple cycles have better potential to contain positive flows since they contribute to both the shortest path p^{k*} and the key path $key(k,i)$. Each simple cycle corresponds to a *simple column* in the RELAX(i) formulation. Suppose p^{k*} and $key(k,i)$ form $\widetilde{n}$ simple cycles. Instead of adding only one column, they add all these $\widetilde{n}$ simple columns at once. Such heuristics do speed up the LP solution procedures.

We will exploit this key variable concept in Section 6.2.

2.3.4 Other price-directive methods

The primal column generation techniques correspond to the dual cutting-plane generation schemes which increasingly refine the polyhedral approximation of the epigraph of the Lagrangian $L(\pi)$. Goffin et al. [137] propose a specialized *analytic center cutting plane method* (ACCPM) [136, 18] to solve the nonlinear min-cost MCNF problems.

Using a piecewise continuous, smooth, convex and nonseparable *linear-quadratic penalty* (LQP) function to penalize the overflow of bundle constraints for each arc, Pinar and Zenios [266] design a parallel decomposition algorithm which parallelizes the procedures of solving

the nonlinear master problem and independent subproblems in the framework of price-directive methods.

Schultz and Meyer [283] give a price-directive interior-point method using a barrier function (see Section 2.6). Many approximation algorithms are also based on price-directive techniques and will be discussed in Section 2.5. The Section 1.3 of Grigoriadis and Khachiyan [160] cites other old price-directive methods that we omit in this section.

2.4 Primal-dual methods (PD)

The *primal-dual method* is a dual-ascent LP solution method which starts with a feasible dual solution, and then iteratively constructs a primal feasibility subproblem called the *restricted primal problem* (RPP) based on the complementary slackness (CS) conditions. It uses the optimal dual solution of RPP to improve the current dual solution if primal infeasibility still exists. The algorithm terminates when all primal infeasibility disappears.

Jewell [182, 181] proposes a PD algorithm to solve the min-cost MCNF problem. Given feasible dual variables, he uses the CS conditions to construct a restricted network which can be viewed as $|K|$ layers of single-commodity networks plus some connecting arcs between layers representing the bundle constraints. The RPP becomes a max MCNF problem. He uses a technique similar to the resource-directive methods which first identifies augmenting paths inside each layer until the layer is saturated, and then solves a capacity reallocation problem between layers. His approach requires enumeration of all paths and cycles, which is not efficient.

Barnhart and Sheffi [32, 38] present a network-based primal-dual heuristic (PDN) which solves a large-scale MCNF problem that is impractical to solve using other exact solution methods. In each primal-dual iteration of PDN, the RPP is solved by a network-based *flow adjustment algorithm* (FAA) instead of using conventional LP methods. When the FAA experiences insufficient progress or cycling, the simplex method will be invoked to solve the RP.

Given a feasible dual solution, PDN iteratively identifies an admissible set S^k which contains arcs with zero reduced cost in the node-arc form for each commodity k, searches for

a flow assignment that satisfies the CS conditions using the FAA, and improves the current dual solution if the RPP is solved with positive primal infeasibility. PDN terminates when either the RPP cannot be solved by the FAA or optimality is achieved. In the first case, if the simplex method is invoked but fails due to the huge size of RPP, Barnhart uses a primal solution heuristic [33] that generates feasible primal solutions from the known dual feasible solution.

Polak [268] gives a Floyd-Warshall-like algorithm to determine the step length required to improve the dual solution in the PD procedures of solving MCNF problems. More details about PD algorithms will be discussed in Chapter 6.

2.5 Approximation methods

Given a desired accuracy $\epsilon > 0$, an ϵ-*optimal* solution is a solution within $(1 + \epsilon)$ (for minimization problems, or $(1 - \epsilon)$ for maximization problems) of the optimal one. A family of algorithms is called a *fully polynomial time approximation scheme* (FPTAS) when it computes an ϵ-*optimal* solution in time polynomial in the size of problem input and ϵ^{-1}.

Shahrokhi and Matula [287] present the first combinatorial FPTAS for a special case of the max-concurrent flow problem that has arbitrary demands and uniform capacities. The idea is to start with a flow satisfying the demands but not the bundle constraints, and then iteratively reroute flow to approach optimality. They also use an exponential function of the arc flow as the "length" of the arc to represent its congestion, so that their algorithm will iteratively reroute flow from longer paths (more congestion) to shorter paths (less congestion). Using different ways for measuring the congestion on arcs and a faster randomized method for choosing flow paths, Klein et al. [205] further improve the theoretical running time.

Leighton et al. [218] use the same randomized technique of [205] but a different rerouting scheme to solve a max-concurrent flow problem that has nonuniform integer capacities. Instead of only rerouting a single path of flow as in [287, 205], they solve a min-cost flow problem which takes more time but reroutes more flows and makes greater progress at each iteration. Goldberg [138] and Grigoriadis and Khachiyan [157] independently use different

randomized techniques to reduce the randomized algorithmic running time of [218] from a factor of ϵ^{-3} to a factor of ϵ^{-2}. Radzik [269] achieves the same complexity using a deterministic approximation algorithm.

Most of the approximation algorithms mentioned above are designed for max MCNF or max-concurrent flow problems, which can be viewed as packing LPs. In particular, let $\hat{P}^k$ denote the polytope of all feasible flows for commodity k. Then, the max MCNF problem can be viewed as packing feasible paths from the polytope $\bigcup_k \hat{P}^k$ subject to the bundle constraints. Similarly, the min-cost MCNF problem can be viewed as packing paths subject to bundle and budget constraints. It can be solved by iterations of a separation problem that determines the optimal budget using bisection search [267].

Recently, Grigoriadis and Khachiyan[157, 158], Plotkin et al. [267], Garg and Könemann [128, 208], and Young [309] have investigated approximation methods for solving packing and covering problems, especially on applications in the MCNF fields. These algorithms are based on Lagrangian relaxation and can be viewed as randomized potential reduction methods [157, 158]. For example, given a current flow f_1, [267] computes an optimal flow f_1^* (or [157] computes an approximately optimal flow) that uses a "penalty" potential function as the arc length to represent the violation of bundle constraints, and updates the flow $f_2 = (1-\sigma)f_1 + \sigma f_1^*$ using some predefined constant σ so that the total "potential" in the next iteration will be reduced. The length for each arc is usually an exponential function [157, 158, 267, 309] (or a logarithmic function [159]) measuring the congestion (i.e., violation of bundle constraints). While [267, 157] improve the potential by a fixed amount and reroute flows at each iteration, [309] gives an "oblivious" rounding algorithm which builds the flow from scratch without any flow rerouting operations. The running time of [267] and [309] depend on the the maximum possible "overflow" of the bundle constraints for points in the polytope $\bigcup_k \hat{P}^k$, denoted as the *width* of the problem.

Karger and Plotkin [192] combine the approximate packing techniques of [267] and the flow rerouting order of [269] to give a better deterministic approximation algorithm to solve min-cost MCNF problems. They develop new techniques for solving the "packing with budget" problems. Such techniques can be used to solve problems with multiple

cost measures on the arc flows. Its direct implementation is slow, but a fast practical implementation is proposed by Oldham et al. [256, 139] using techniques from [159, 192, 267] to fine-tune many algorithmic parameters.

Garg and Könemann [128, 208] propose a simple combinatorial approximation algorithm similar to Young's [309], which has the most improvement at each iteration using shortest path computations. Fleischer [111] further improves the algorithms of [128] and gives approximation algorithms that have the best running time for solving the max MCNF, max-concurrent flow and min-cost MCNF problems.

Grigoriadis and Khachiyan [158] use a 2-phase exponential-function reduction method to solve general block-angular optimization problems. Both phases are resource-sharing problems. The first phase is to obtain a feasible solution interior to the polytope $\bigcup_k \hat{P}^k$. The second phase uses a modified exponential potential function to optimize on both bundle and flow conservation constraints.

Awerbuch and Leighton [19, 20] give deterministic approximation algorithms which use local-control techniques similar to the preflow-push algorithm of Goldberg and Tarjan [141, 143]. Unlike previous approximation algorithms that usually relax the bundle constraints and then determine shortest paths or augmenting paths to push flow towards sinks, their algorithms relax the flow conservation constraints and use an "edge-balancing" technique which tries to send a commodity across an arc (u, v) if node u has more congestion than node v. Based on these edge-balancing methods [19, 20] (denoted as *first-order algorithms*), Muthukrishnan and Suel [248] propose a *second-order distributed flow algorithm* which decides the amount of flow to be sent along an arc by a combination of the flow sent on the same arc in previous iteration and the flow determined using the first-order algorithms. Their experiments show a significant improvement in running time compared to the first-order algorithms. Kamath et al. [190, 262] generalize the algorithm of [20] to solve min-cost MCNF problems, but their algorithm is inferior to [192].

Schneur and Orlin [280, 281] present a penalty-based approximation algorithm that solves the min-cost MCNF problems as a sequence of a finite number of scaling phases. In each phase, $|K|$ SCNF problems are solved, one for each commodity. Besides the original

linear flow cost $c_{ij}^k x_{ij}^k$, they assign a quadratic penalty term ρe_{ij}^2 for each arc (i,j) where $e_{ij} = \max\{\sum_{k\in K} x_{ij}^k - u_{ij}, 0\}$ represents the overflow (or excess) on arc (i,j). Thus each SCNF problem becomes a nonlinear min-cost network flow problem which is solved by a negative cycle canceling algorithm. In particular, at each scaling phase their algorithm iteratively sends exactly δ units of flow along negative δ-cycles which are defined as a cycle that allows flow of δ units. In the end of a scaling phase, the flow x is (δ, ρ)-*optimal* which means there exists no negative cycle that allows flow of δ units. By using new δ (usually $\frac{\delta}{2}$) and ρ (usually $R\rho$ where $1 < R < 2$) in the next scaling phase, the algorithm can be shown to terminate in finite number of scaling phases.

Bienstock [48] solves network design problems and other MCNF test problems (from [139]) by an approximate algorithm based on [158, 267] with new lower bounding techniques. Instead of using specialized codes such as shortest path or min-cost network flow programs, he uses CPLEX 6.5 to solve the subproblems. In his recent comprehensive paper [49], he reviews many exponential potential function based approximation algorithms and discusses detailed issues regarding empirical speedups for algorithms based on this framework.

The following summarizes the current fastest FPTAS :

- max MCNF problems: [111]
- max-concurrent flow problems: [111] (when graph is sparse, or $|K|$ is large), [218] and [269]
- min-cost MCNF problems: [111] (when $|K| > \frac{|A|}{|N|}$), [159]

2.6 Interior-point methods

Kapoor and Vaidya [191] speed up Karmarkar's algorithm [193] for solving MCNF problems. Using special computing and inversion techniques for certain matrices, Vaidya's path-following interior-point algorithm [300] achieves one of the current fastest running times for obtaining exact MCNF solutions. Murrary [245] also gives a specialized MCNF algorithm based on the algorithm of Monteiro and Adler [242].

Putting the flow imbalance as a quadratic term (equal to 0 if and only if flow conservation

is maintained) in the objective function, Kamath and Palmon [189, 262] present an interior-point based FPTAS that solves a quadratic programming (QP) min-cost MCNF problem subject to only the bundle and nonnegativity constraints. With a sufficiently small ϵ, their algorithm can compute an exact solution using the rounding algorithm of Ye [307].

While these algorithms introduced above have good theoretical time bounds, they are considered to be practically unattractive. We now introduce some practical interior-point MCNF algorithms.

Schultz and Meyer [283] develop a decomposition method based on solving logarithmic barrier functions. Their algorithm involves three phases: relaxed phase, feasibility phase, and refine phase. After obtaining a solution of the relaxed problem (i.e., relaxing the bundle constraints), the feasibility phase tries to obtain a feasible interior point which will be refined to optimality in the refine phase. Both the feasibility and refine phases find an approximate solution of the *barrier problem* using a generalization of the *Frank-Wolfe* method [120] which does a multi-dimensional search rather than a line search and is more easily parallelized. Instead of using a complex coordinator [283] to determine step lengths in the search directions, Meyer and Zakeri [239, 310] further improve this procedure using multiple simple coordinators that take more advantage of parallelism.

General interior-point methods for LP usually need to solve a symmetric system of linear equations. In many cases, the Cholesky factorization still creates a dense submatrix even if the best fill-in minimization heuristics are applied. To speed up the factorization by taking advantage of the MCNF structure, basis partitioning techniques are used by Choi and Goldfarb [76] and Castro [67] in their primal-dual interior-point algorithms. To efficiently solve the dense linear system, Choi and Goldfarb [76] suggests parallel and vector processing while Castro [67] applies a *preconditioned conjugate gradient* (PCG) method. Parallel implementation on primal-dual interior-point methods using PCG techniques can be found in Castro and Frangioni [69] and Yamakawa et al. [305].

A specialized *dual affine-scaling* method (DAS) for solving an undirected min-cost MCNF problem has been implemented by Chardaire and Lisser [72, 73]. Starting with a feasible dual interior point, the algorithm iteratively searches for an improving direction

by solving a symmetric linear system (similar to [76, 67]) that approaches the optimal dual solution. Using the same partitioning techniques of [76, 67], they solve the linear system without any special techniques. The same authors also implement a specialized analytic center cutting plane method (ACCPM) [136], a cutting plane method previously introduced in Section 2.3.4 which can be viewed as an interior-point method since it computes the central point of a polytope in each iteration.

2.7 Convex programming methods

Although this dissertation concentrates on solving the linear MCNF problems, here we introduce some nonlinear convex programming techniques in the literature ([90] in Section 2.7.2 and [188] in Section 2.7.3 are for LPs). Most of them are related to *augmented Lagrangian* methods which find a saddle point of a smooth min-max function obtained by augmenting the Lagrangian with quadratic terms of both primal and dual variables.

2.7.1 Proximal point methods

Based on their previous work [178], Ibaraki and Fukushima [179] apply a primal-dual proximal point method to solve convex MCNF problems. The method identifies an approximate saddle point of the augmented Lagrangian at each iteration and guarantees that these points converge to the unique Kuhn-Tucker point of the problem. Similar methods are also developed by Chifflet et al. [75] and Mahey et al. [227].

2.7.2 Alternating direction methods (ADI)

De Leone et al. [90] propose three variants of *alternating direction methods* (ADI) [209] which may be viewed as block Gauss-Seidel variants of augmented Lagrangian approaches that take advantage of block-angular structure and parallelization. These methods take alternating steps in both the primal and the dual space to achieve optimality. In particular, ADI is designed to solve the following problem:

$$\begin{array}{lrcl} \min & G_1(x) & + & G_2(z) \\ s.t. & Cx + b & = & Dz \end{array} \qquad \text{(ADI)}$$

For example, let x be the flow vector, z be the proximal vector that reflects the consumption of the shared capacity, and p be the associated dual variables. The *resource proximization method* (RP_ADI) uses the following settings: $G_1(x) := \sum_{k\in K}\sum_{(i,j)\in A} c_{ij}^k x_{ij}^k$; $G_2(z) := 0$ if $\sum_{k\in K} z_{ij}^k \le u_{ij}\ \forall (i,j)\in A$, or $G_2(z) := \infty$, otherwise; $C := I_{|A||K|}$; $D := I_{|A||K|}$; $b := 0$ and Λ is a diagonal penalty matrix whose k^{th} diagonal entry is $\Lambda_k \in (0,\infty)$. Define the augmented Lagrangian

$$L_\Lambda^k = \min_{\substack{\widetilde{N}x^k=b^k \\ 0\le x_{ij}^k\le u_{ij}}} \sum_{(i,j)\in A} ((c_{ij}^k + p_{ij}^k)x_{ij}^k + \frac{1}{2}\Lambda_k(x_{ij}^k - z_{ij}^k)^2)$$

where $\widetilde{N}$ is the node-arc incidence matrix. Starting with a suitable dual variable $p_0 \ge 0$ and an initial capacity allocation z_0 satisfying $\sum_{k\in K} z_0^k = u$, RP_ADI iteratively computes the minimizer x^{k*} of L_Λ^k for each commodity k, updates multiplier p, adjusts the capacity allocation r, and updates the penalty matrix Λ of L_Λ^k until the termination criterion is met. Although RP_ADI is guaranteed to achieve the optimal objective value, primal feasibility may not be satisfied.

They also propose two other methods, the *activity and resource proximization method* (ARP_ADI) and the *activity proximation method* (AP_ADI). Their computational results show that AP_ADI and RP_ADI run significantly faster than ARP_ADI, and all of them are faster than the general-purpose LP solver MINOS 5.4 [247]. A similar method is also developed by Eckstein and Fukushima [99].

2.7.3 Methods of games

Kallio and Ruszczyńsky [188] view the solution procedure to a LP as a nonzero-sum game with two players. In particular, the primal players minimize the augmented Lagrangian function for the primal problem and the dual players maximize the augmented Lagrangian function for the dual problem. Each player optimizes his own objective by assuming the other's choice is fixed. They propose a parallel method where processors carry out under-relaxed Jacobi steps for the players. Based on the same idea of [188], Ouorou [257] solves *monotropic programming problems* [271] which minimize a separable convex objective function subject to linear constraints. He solves convex min-cost MCNF problems using a

block-wise Gauss-Seidel method to find an equilibrium of the game in a primal-dual algorithmic framework. These techniques are similar to the alternating direction method discussed previously.

2.7.4 Other convex and nonlinear programming methods

Nagamochi et al. [249] use the relaxation method of Bertsekas [43, 46], a specialized PD algorithm, to solve strictly convex MCNF problems.

Lin and Lin [222] solve quadratic MCNF problems by a projected Jacobi method. They introduce a new *dual projected pseudo-quasi-Newton* (DPPQN) method to solve the quadratic subproblems induced in the projected Jacobi method. The DPPQN method computes a fixed-dimension sparse approximate Hessian matrix, which overcomes the difficult computation of an indefinite-dimension dense Hessian matrix that arises in the conventional Lagrangian Newton method.

Ouorou and Mahey [258] use a minimum mean cycle cancelling based algorithm [142] to solve a MCNF problem with a nonlinear separable convex cost function. More techniques about solving nonlinear convex MCNF problems are introduced in Chapter 8 and Chapter 9 of [45]. For survey on algorithms for nonlinear convex MCNF problems, see Ouorou et al. [259].

2.8 Methods for integral MCNF problems

All the algorithms introduced before this section are fractional solution methods. Adding the integrality constraints makes the MCNF problems much harder.

Much research has been done regarding the characterization of integral MCNF problems. For example, Evans et al. [106] have given a necessary and sufficient condition for unimodularity in some classes of MCNF problems. Evans makes the transformation from some MCNF problems to their equivalent uncapacitated SCNF problems [100, 101, 103], and proposes a specialized simplex algorithm [102] and heuristics [104, 105] for certain MCNF problems. Truemper and Soun [298, 299, 292] further investigate the topic and obtain additional results about unimodularity and total unimodularity for general MCNF

problems. Other research about the characteristics of quater-integral, half-integral and integral solutions of MCNF problems can be found in [224, 282, 197, 198].

Aggarwal et al. [1] propose a resource-directive heuristic which first determines an initial feasible integral capacity allocation for each commodity on each arc, and then performs parametric analysis by varying the right-hand-side and using the corresponding dual variables to solve knapsack-type problems which determine better allocation for the next iteration. Their methods produce good integral feasible solutions but may require many LP computations for the parametric analysis, and thus may not be suitable for large problems.

The general integral MCNF algorithms are usually based on LP-based branch-and-bound schemes such as *branch-and-cut* [50, 57] or *branch-and-price* [34, 35, 13]. In branch-and-cut, cuts (if possible, facet-inducing ones) are dynamically generated throughout the branch-and-bound search tree to cut off fractional solutions. On the other hand, branch-and-price [302, 37] generates variables that are used in conjunction with column generation to strengthen the LP relaxation and resolve the symmetry effect due to formulations with few variables. Barnhart et al. [34, 35] apply *branch-and-price-and-cut* which generates both variables and cuts during the branch-and-bound procedures to solve binary MCNF problems in which integral flow must be shipped along one path for each commodity (OD pair). Alvelos and Carvalho [13] propose another branch-and-price algorithm to solve the general integral MCNF problems which allows integral flow to be shipped along several paths for each commodity (OD pair).

2.9 Heuristics for feasible LP solutions

Occasionally, there are problems that need fast and good feasible primal or dual solutions. Barnhart proposes a dual-ascent heuristic [33] to quickly generate a good feasible dual solution based on the CS conditions (see Section 2.3.2) Suppose σ^* is the optimal dual variable for (2.1) and $-\pi^*$ is the optimal dual variable for the bundle constraint (2.2). Then, (CS.3) implies that the optimal flow should be assigned along shortest paths using $c_a + \pi_a^*$ as the new arc length for arc a, and (CS.1) implies that $\pi_a^* > 0$ when arc a is saturated and $\pi_a^* = 0$ when a is under-capacitated. Thus Barnhart's heuristic increases π_a

when a shortest path demand exceeds its capacity u_a for arc a, and decreases π_a ($\pi_a > 0$) when a shortest path demand is less than u_a.

The same paper also gives a primal solution generator based on the observation from (CS.1) that in the optimal solution an arc with a positive dual price should be saturated. So, the heuristic will try to send flow along the shortest path that contains arcs with positive dual prices first, and then along the arcs with zero dual prices. In general, this primal solution generator does not guarantee a feasible primal solution, but it works well in all of their tests.

2.10 Previous computational experiments

Comprehensive survey papers on solution methods and computational results for MCNF problems were done more than two decades ago by Assad [15, 17], Ali et al. [10], and Kennington [201]. All of these surveys suggest that the price-directive methods (DW) outperform the resource-directive methods, which are in turn superior to the basis-partitioning methods. A brief summary of the computational experiments done before 1990 can also be found in Farvolden et al. [109].

There are some popular MCNF test problems used by many researches for evaluating efficiency of their algorithms. For example, the Mnetgen family (mnet) created by Ali and Kennington [11], and the PDS family (PDS) from Carolan et al. [62] which models a patient distribution system where decisions have to be made in evacuating patients away from a place of military conflict, are commonly tested. The size of the PDS family depends on the planning horizon since it is a time-space network.

Chardaire and Lisser [73] implement four algorithms to solving undirected min-cost MCNF problems up to size $|N| = 119$, $|A| = 302$, $|K| = 7021$: (1) a simplex method using the basis-partitioning technique from [202], (2) a dual affine scaling method (DAS), (3) DW whose master problem is solved by a basis partitioning technique, and (4) an analytic center cutting plane method (ACCPM) [136]. They compare these four methods with CPLEX 4.0. The results show that DW is the most efficient implementation, DAS is the slowest in general, the basis partitioning method performs better than the DAS for smaller

cases, and CPLEX 4.0 performs well for smaller cases but worse for larger cases. Similar undirected message routing problems up to $|N| = 100$, $|A| = 699$, $|K| = 1244$ are solved by McBride and Mamer [237] who compare (1) EMNET (see Section 2.1), (2) EMNET using shortest path pricing, and (3) CPLEX 6.5. Their results show that EMNET using shortest path pricing improves the generic EMNET, which in turn is much faster than CPLEX 6.5. A similar conclusion is also drawn in another paper by the same authors [229] which tests more MCNF problems such as PDS, mnet, dmx and JLF (which are all available at the website maintained by Antonio Frangioni[1]).

McBride [234] summarizes the computational performance of several MCNF algorithms (on different platforms), including (1) decomposition methods [266, 311],[283], (2) interior point methods [62],[225],[231], (3) a primal simplex method with advanced basis [236], and (4) a simplex method using basis-partitioning technique (EMNET) [233], on specific test problem sets such as KEN and PDS from the NETLIB library. He concludes that the basis-partitioning approach is efficient, especially in postprocessing which is often required in practice. However, the interior-point approaches can take more advantage of multiple processors operating in parallel.

Castro [67] shows that his specialized interior-point method, IPM, is faster than CPLEX 4.0, which in turn is much faster than PPRN [70]. Castro [66] discusses the empirical running times of some recent algorithms that have appeared in the last decade, compared with CPLEX 6.5. In basis-partition codes, EMNET [235] performs similarly to CPLEX 6.5 but better than PPRN [70] for PDS test problems. In price-directive codes, bundle methods [119] perform similarly to CPLEX 6.5.

Solving the QP test problems from the mnet and PDS families with size up to 500,000 variables and 180,000 constraints, Castro [68] concludes that IPM [67, 68] performs up to 10 times faster than the CPLEX 6.5 barrier solver. Both IPM and the CPLEX 6.5 barrier solver are much faster than PPRN [70], which in turn is much faster than ACCPM [137].

Despite their excellent theoretical running times, only few approximation algorithms have been implemented for practical use. [158], [139, 256], and [48] have reported their

[1] http://www.di.unipi.it/di/groups/optimize/Data/MMCF.html

computational experiments. MCMCF, the approximation code in [139, 256], is shown to be a few magnitudes faster than CPLEX 4.0.9 (dual simplex method) and PPRN [70] on problems generated by RMFGEN, MULTIGRID, and TRIPARTITE generators (see [139, 256]). These three problem generators are also used in [48].

Schneur and Orlin [281] compare their scaling algorithm (SAM.M) with other codes DW, PDN, and PPLP (DW and PDN are from [32], PPLP is from [109]). Their experiments show that PPLP and SAM.M are faster than DW and PDN on problems up to $|N| = 85$, $|A| = 204$, $|K| = 18$.

Farvolden et al. [109] compare their basis-partitioining method, PPLP, with MINOS and OB1 [226] on large-scale LTL networks and show that PPLP outperforms the others by a large factor.

In [32, 38], Barnhart compares her algorithm PDN to conventional DW decomposition and two primal-dual (PD) algorithms. It is shown that PDN is more advantageous for large-scale MCNF problems. In particular, her heuristics can produce good solutions for some freight assignment problems which the other LP-based methods (DW, PD) can not solve. Barnhart [33] also develops heuristics which show promising performance in obtaining a fast and good solution compared to OB1 and OSL [97].

Barnhart et al. [36] experiment with their PATH and CYCLE-RELAX algorithms (they use the formulation (P_PATH) and (RELAX(i)) as introduced in Section 2.3.3)) on message routing problems which contain many OD commodities. The CYCLE-RELAX algorithm is much faster than OB1 and OSL on solving problems up to $|N| = 324$, $|A| = 1019$, $|K| = 150$. When compared with their PATH algorithm, which is the conventional DW decomposition using column generation, the CYCLE-RELAX algorithm also displays faster running times on problems up to $|N| = 500$, $|A| = 1300$, $|K| = 5850$. In their later papers [34, 35], they solve binary MCNF problems up to $|N| = 50$, $|A| = 130$, $|K| = 585$. Their simple cycle heuristic that generates all the simple cycles from the symmetric difference between the shortest path and key path has shown to be very effective. In particular, the time to solve the LPs and the number of iterations of generating columns have been reduced by an average of 40% and 45% respectively, but the total number of columns generated is

not increased. This means the simple cycle heuristic helps to generate good columns in fewer iterations, which shortens the overall computational time. Therefore, we will also incorporate this simple cycle heuristic in our proposed algorithm in Chapter 6.

2.11 Summary

We have introduced and summarized most of the methods in the literature for solving the MCNF problems. Different solution techniques may be advantageous for different MCNF problems depending on the problem characteristics. This dissertation focuses on solving the class of MCNF problems in which many OD commodities have to be routed. Therefore, the algorithms by Barnhart et al. [36, 34, 35] seem to be most suitable for our purpose. In Chapter 6, we will propose new primal-dual column generation algorithms which follow the PD algorithmic framework. We will also exploit the ideas from the CYCLE-RELAX algorithm.

In our algorithms (see Chapter 6), subproblems which seek shortest paths between multiple pairs must be repeatedly solved. Therefore, efficient multiple pairs shortest paths (MPSP) algorithms will speed up the overall computational efficiency of our MCNF algorithms. We introduce several shortest path algorithms in Chapter 3, give our own new MPSP algorithms in Chapter 4, and describe their computational performance in Chapter 5. We will introduce our MCNF algorithms and their computational performance in Chapter 6.