A Computationally Efficient Method for Nonlinear Multicommodity Network Flow Problems

Shin-Yeu Lin, Ch'i-Hsin Lin

Department of Control Engineering, National Chiao Tung University, Hsinchu, Taiwan

Received 13 July 1994; accepted 12 December 1996

Abstract: In this paper, we present a new method for solving nonlinear multicommodity network flow problems with convex objective functions. This method combines a well-known projected Jacobi method and a new dual projected pseudo-quasi-Newton (DPPQN) method which solves multicommodity flow quadratic subproblems induced in the projected Jacobi method. The DPPQN method is a dual Newton-type method that differs very much from the conventional Lagrangian Newton method; our method fully exploits the structural advantages of network-type linear equality constraints to obtain a constant sparse approximate Hessian matrix with a decoupling structure and includes a novel finite-iteration successive projection and (truncated) seal algorithm to resolve the difficulty caused by coupling capacity constraints. The DPPQN method also consists of two decomposition effects, the commodity decomposition effect and the arc decomposition effect, which resolve the potential numerical difficulties caused by large dimensions. We show the convergence of our method including the convergence of the finite-iteration successive projection and (truncated) seal algorithm. Compared with the Frank–Wolfe with PARTAN algorithm in which a price-directive decomposition method is used to solve linearized multicommodity flow problems, our method is dramatically faster in terms of the CPU time on a Sparc-10 workstation at solving numerous nonlinear multicommodity network flow examples. © *1997 John Wiley & Sons, Inc.* Networks **29:** 225–244, 1997

Keywords: nonlinear multicommodity flow; optimization methods; decomposition; projection; sparse matrix technique

1. INTRODUCTION

In this paper, we consider a nonlinear multicommodity network flow problem (NMNFP) of the following form [1]:

$$\min \sum_{a=1}^{m} Z_a(f_a), \tag{1a}$$

subject to

$$E_r f^k = b_r^k, \quad k = 1, \ldots, K \tag{1b}$$

$$f^k \geq 0, \quad k = 1, \ldots, K \tag{1c}$$

$$\sum_{k=1}^{K} f_a^k \leq d_a, \quad a = 1, \ldots, m, \tag{1d}$$

where the notations adopted from [1] are defined based on a directed graph $G = (\mathcal{N}, \mathcal{A})$, so that a denotes the index of an arc in the set $\mathcal{A}$; m, the total number of arcs of $\mathcal{A}$; k, the index of a commodity; and K, the total number of commodities; $f^k = (f_1^k, \ldots, f_m^k)$, with f_a^k being the flow of commodity k on arc a; $f_a = \sum_{k=1}^{K} f_a^k$

Correspondence to: S.-Y. Lin; e-mail: sylin@cc.nctu.edu.tw
Contract grant sponsor: National Science Council (Taiwan); contract grant number: NSC83-0404-E009-115

denotes the total flow on arc a, and E, the node–arc incidence matrix of G. Letting F^k denote the value of the flow of commodity k to be sent from source node s to destination node t, then $b^k = (b_1^k, \ldots, b_n^k)$ with

$$b_i^k = \begin{cases} F_i^k & \text{if } i = s, \\ -F_i^k & \text{if } i = t, \\ 0 & \text{otherwise} \end{cases},$$

where i denotes the index of a node in the set $\mathcal{N}$ and n is the total number of nodes in $\mathcal{N}$. Thus, $Ef^k = b^k$ denotes the flow conservation law of commodity k. Since $Ef^k = b^k$ consists of one redundant equation that can be deleted to form $E_r f^k = b_r^k$, *where* E_r is formed by deleting one row from E and b_r^k is formed by deleting the corresponding component from b^k. Without loss of generality, we may delete the nth equation. $f^k \geq 0$ denotes the nonnegativity condition on flow. We let $d = (d_1, \ldots, d_m)$, with d_a being the capacity of arc a, then $\sum_{k=1}^{K} f_a^k \leq d_a$ denotes the capacity constraint on the total flow of arc a. The arc cost $Z_a(f_a)$ is a convex function of f_a; therefore, the objective function $\sum_{a=1}^{m} Z_a(f_a)$, the sum of all arc costs, is convex.

Such NMNFPs with convex objective functions are usually raised in traffic-assignment problems [9, 20, 23, 26] and data network routing problems [6, 11, 12, 30]. Solution techniques for NMNFPs have mostly originated from nonlinear programming algorithms and have been specialized to exploit the structure of the linear constraints, e.g., the feasible direction approach [17], linear approximation algorithms [12, 14, 18, 19, 24], the equilibration approach [8, 9, 20, 29], reduced gradient method [20, 24], the gradient projection method [27], and certain other methods [7, 13, 15].

All the above-mentioned methods take the primal approach. Recently, Bertsekas et al. [4] and Ohuchi and Kaji [25] used dual-relaxation methods to solve the nonlinear minimum-cost single-commodity flow problem. Nagamochi et al. [22] presented a dual-relaxation method for the nonlinear multicommodity flow problem with a strictly convex objective function, and they considered the coupling capacity constraints (1d). These dual methods [4, 22, 25] are gradient-type methods. To develop an efficient Newton-type approach to the NMNFP is not an easy task because the coupling capacity constraints (1d) may induce a dense Hessian matrix that will cause severe computational inefficiency; in fact, the difficulty resulting from the coupling capacity constraint complications has been bypassed in some of the recently published efficient methods [3, 10, 28] by authors who ignored the (explicit) coupling capacity constraints (1d) in their formulation.

In this paper, we present a new dual Newton-type method—the *dual projected pseudo-quasi-Newton* (*DPPQN*) *method.* This method differs from the conventional Lagrangian Newton method in the way it handles the inequality constraints, (1c) and (1d). The latter employs a Lagrange multiplier μ associated with the inequality constraints that may cause a dense and indefinitely dimensioned Hessian matrix because of the presence of coupling capacity constraints (1d) and the change of indices for the active set of inequality constraints. Our DPPQN method treats the inequality constraints, (1c) and (1d), as the primal variable domain in the dual function. It was this trick that made it possible for us to obtain a fixed-dimension sparse approximate Hessian matrix. However, to calculate the gradient of the dual function in each iteration of the DPPQN method, we need to solve for the optimal dual-function primal variables, for which we developed *a two-phase method.* The first phase solves the dual-function primal variables by relaxing the inequality constraints on the primal variables. The second phase projects the solution of primal variables obtained in phase 1 onto the set of inequality constraints. To achieve the projection required in the second phase, we developed an efficient *K-iteration successive projection and* (*truncated*) *seal algorithm.* This efficient two-phase method overcomes complications caused by the coupling capacity constraints. Therefore, the proposed DPPQN method is a Newton-type method that fully exploits (i) the structure of network-type linear equality constraints to obtain a sparse approximate Hessian matrix of fixed dimensions, and (ii) the special type of multicommodity flow coupling capacity constraints, to develop an efficient K-iteration successive projection and (truncated) seal algorithm. In addition, our method also achieves two decomposition effects, commodity decomposition and arc decomposition, which also contribute to its computational efficiency and resolve the potential numerical difficulties caused by large dimensions.

However, the DPPQN method cannot be applied directly to solve the NMNFP formulated in (1a)–(1d) because the objective function that we consider is generally not a strictly convex function as in [22]. Therefore, we have to combine our method with the well-known projected Jacobi method, which provides a strictly convex objective function for the multicommodity flow quadratic subproblem and makes the sparse approximate Hessian matrix of the dual function a constant sparse matrix with a decoupling structure, enhancing the merit of the DPPQN method. Throughout the paper, we make the following two assumptions about the objective function $Z(f) \equiv \sum_{a=1}^{m} Z_a(f_a)$:

Assumptions

1. $Z(f)$, as well as $Z_a(f_a)$, are convex and bounded from below for every feasible f.

2. $Z(f)$ is twice continuously differentiable in the interior of the feasible set of f.

We say that f is feasible if f satisfies (1b), (1c), and (1d); furthermore, the set of all feasible f is called the feasible set of f. The assumptions are easily satisfied in most of the practical applications [6, 9, 11, 12, 20, 23, 26, 30] or problems involving quadratic objective functions [10]. If we apply the mean-value theorem, Assumption 2 implies that a Lipschitz constant K_1 exists such that the Lipschitz condition is satisfied; i.e., $\|\nabla_f Z(f') - \nabla_f Z(f'')\|_2 \le K_1\|f' - f''\|_2$, for every feasible f' and f''.

2. PROJECTED JACOBI METHOD

The projected Jacobi method uses iterations of the following form to solve the NMNFP (1a)–(1d):

$$f(l+1) = f(l) + \alpha(l)df^*(l), \tag{2}$$

where l denotes the index of iteration, $\alpha(l) > 0$ is a step-size, and $df^*(l) = (df_1^*(l), \ldots, df_m^*(l))$ is the vector of increment of f that solves the following multicommodity flow quadratic subproblem:

$$\min \sum_{a=1}^{m} [\tfrac{1}{2}df_a^T H_a(l)df_a + \nabla_{f_a} Z_a^T(l)df_a], \tag{3a}$$

subject to

$$E_r(f^k(l) + df^k) = b_r^k, \quad k = 1, \ldots, K \tag{3b}$$

$$f^k(l) + df^k \ge 0, \quad k = 1, \ldots, K \tag{3c}$$

$$\sum_{k=1}^{K} f_a^k(l) + df_a^k \le d_a, \quad a = 1, \ldots, m, \tag{3d}$$

where $Z_a(l)$ denotes $Z_a(f_a(l))$, the $K \times K$ matrix $H_a(l)$ is a diagonal matrix such that the kth diagonal element

$$h_a^k(l) = \begin{cases} \dfrac{\partial^2 Z_a(l)}{\partial f_a^{k^2}} & \text{if } \dfrac{\partial^2 Z_a(l)}{\partial f_a^{k^2}} \ge \gamma, \\ \gamma, & \text{otherwise,} \end{cases} \tag{4}$$

where γ is a small positive real number, $\nabla_{f_a} Z_a(l)$ denotes the gradient of Z_a with respect to f_a evaluated at $f_a(l)$, T denotes the transpose, and $df_a = (df_a^1, \ldots, df_a^K)$.

In general, a basic requirement for a method that solves a minimization (or maximization) problem is that the method be a descent (or ascent) method. This requirement ensures the existence of a point with a smaller (or larger) objective value along the descent (or ascent) direction. Starting from a feasible f, proof that the projected Jacobi method is a descent method and that converges under certain conditions has been given in [5, Proposition 3.7]; we restate it here as a lemma.

Lemma 1. *Starting from a feasible f, if there exists a* $\gamma > 0$ *such that*

$$(f_a' - f_a'')^T H_a(l)(f_a' - f_a'') \ge \gamma\|f_a' - f_a''\|_2^2, \tag{5}$$

for any $f_a', f_a'' \in \Re^K$, *every arc a and every l, and if the step-size* $\alpha(l)$ *is sufficiently small, then the projected Jacobi method* (2) *is a descent method and any limit point of the sequence* $\{f(l)\}$ *generated by* (2) *will solve* (1*a*)–(1*d*).

In the above sufficient conditions, (5) is satisfied by the definition of $H_a(l)$ given in (4); however, the sufficiently small $\alpha(l)$ will cause ambiguity in practice. To cope with this difficulty, we can further develop the following formula based on the proof in [5] that shows (2) is a descent method; i.e., if $0 < \alpha(l) < [(2\gamma)/K_1]$, then

$$\begin{aligned} &Z(f(l) + \alpha(l)df^*(l)) \\ &\quad < Z(f(l)) - \alpha^2(l)\left(\frac{\gamma}{\alpha(l)} - \frac{K_1}{2}\right)\|df^*(l)\|_2^2, \end{aligned} \tag{6}$$

which provides a range of $\alpha(l)$ to ensure the descent property of (2). However, we need to restrict $\alpha(l) \le 1$ to ensure the feasibility of $f(l) + \alpha(l)df^*(l)$ when $f(l)$ is feasible. Thus, we have the following lemma:

Lemma 2. *Starting from a feasible f, if* $H_a(l)$ *is constructed according to* (4), *and if* $0 < \alpha(l) < \min(1, (2\gamma/K_1))$ *for every l, then method* (2) *is a descent method and any limit point of the sequence* $\{f(l)\}$ *generated by* (2) *will solve* (1*a*)–(1*d*).

However, in most of the applications [6, 9, 11, 12, 20, 23, 26, 30], prior knowledge of the Lipschitz constant K_1 is not available. Thus, to ensure convergence, we need to develop an Armijo-type step-size rule to determine the step-size $\alpha(l)$. First, we need to further develop the inequality (6), as follows: If $0 < \alpha(l) < (\gamma/K_1)$, we can derive from (6) that

$$\begin{aligned} &Z(f(l) + \alpha(l)df^*(l)) \\ &\quad < Z(f(l)) - \frac{\gamma}{2}\alpha(l)\|df^*(l)\|_2^2. \end{aligned} \tag{7}$$

Let $0 < \tau_J < 1$. The sequence τ_J^m, $m = 0, 1, 2, \ldots$, is then monotonically decreasing. There must exist a nonnegative integer m such that $0 < \tau_J^m < (\gamma/K_1)$. Thus, the Armijo-type step-size can be determined by increasing m from 0 in increments of 1 and checking whether the following inequality holds:

$$Z(f(l) + \tau_J^m df^*(l)) < Z(f(l)) - \frac{\gamma}{2}\tau_J^m \|df^*(l)\|_2^2. \tag{8}$$

We set $\alpha(l) = \tau_J^{m(l)}$, where $m(l)$ is the smallest nonnegative integer m for which (8) holds, and this $\alpha(l)$ satisfying $0 < \alpha(l) \leq 1$ ensures the descent property of (2) and the feasibility of $f(l+1)$. Then, from Lemma 2, we have the following theorem:

Theorem 1. *Starting from a feasible f, if $H_a(l)$ is constructed according to* (4) *and if $\alpha(l) = \tau_J^{m(l)}$,* where $0 < \tau_J < 1$ *and $m(l)$ is the smallest nonnegative integer m for which* (8) *holds, then method* (2) *is a descent method and any limit point of the sequence $\{f(l)\}$ generated by* (2) *will solve* $(1a)-(1d)$.

From Theorem 1, we see that we do not need any knowledge of the Lipschitz constant K_1 to determine the step-size $\alpha(l)$.

3. THE DUAL PROJECTED PSEUDO-QUASI-NEWTON METHOD

3.1. Preliminaries

Let the set $\mathcal{F}$ denote the set of f that satisfies the inequality constraints on flows, i.e., $f \in \mathcal{F}$ if and only if f satisfies (1c) and (1d), or $f + df \in \mathcal{F}$ if and only if $f + df$ satisfies (3c) and (3d). We see that if both f and $f + df \in \mathcal{F}$ then $f + \alpha df \in \mathcal{F}$ for $0 < \alpha \leq 1$. Using this notation, we may rewrite (3a)–(3d) as

$$\min_{f(l)+df\in\mathcal{F}} \sum_{a=1}^{m} [\tfrac{1}{2} df_a^T H_a(l) df_a + \nabla_{f_a} Z_a^T(l) df_a], \tag{9a}$$

subject to

$$E_r f^k(l) - b_r^k + E_r df^k = 0. \tag{9b}$$

We define the $m \times m$ diagonal matrix $H^k(l)$ to be such that the ath diagonal element of $H^k(l)$ is the kth diagonal element of the $K \times K$ diagonal matrix $H_a(l)$. Thus, $\sum_{a=1}^m [\frac{1}{2} df_a^T H_a(l) df_a + \nabla_{f_a} Z_a^T(l) df_a]$ can be written as $\sum_{k=1}^K [\frac{1}{2} df^{k^T} H^k(l) df^k + \sum_{a=1}^m [\partial Z_a(l)/\partial f_a^k] df_a^k]$. Then, the dual problem of (9a)–(9b) is shown in (10):

$$\max_{\lambda} \phi(\lambda), \tag{10}$$

where the dual function

$$\phi(\lambda) = \min_{f(l)+df\in\mathcal{F}} \sum_{k=1}^{K} \left[\frac{1}{2} df^{k^T} H^k(l) df^k + \sum_{a=1}^{m} \frac{\partial Z_a(l)}{\partial f_a^k} df_a^k\right] + \sum_{k=1}^{K} \lambda^{k^T}[E_r f^k(l) - b_r^k + E_r df^k] \tag{11}$$

is a function of $\lambda = (\lambda^1, \ldots, \lambda^K)$, which is a $K(n-1)$-dimensional vector of Lagrange multipliers and $\lambda^k \in \mathrm{R}^{(n-1)}$ for each k. We order f and df in the following sequences: $f_1^1, \ldots, f_m^1, f_1^2, \ldots, f_m^2, \ldots, f_1^K, \ldots, f_m^K$, and $df_1^1, \ldots, df_m^1, df_1^2, \ldots, df_m^2, \ldots, df_1^K, \ldots, df_m^K$, respectively. Under this ordering of f and df, we let the $K(n-1) \times Km$ block diagonal matrix $\mathcal{E}_r$ be

$$\mathcal{E}_r = \begin{bmatrix} E_r & 0 & \cdots & 0 \\ 0 & E_r & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & E_r \end{bmatrix} \tag{12}$$

and let the $mK \times mK$ diagonal matrix $\mathcal{H}(l)$ be

$$\mathcal{H}(l) = \begin{bmatrix} H^1(l) & 0 & \cdots & 0 \\ 0 & H^2(l) & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & H^K(l) \end{bmatrix}. \tag{13}$$

Then, (9a)–(9b) can be rewritten as

$$\min_{f(l)+df\in\mathcal{F}} \tfrac{1}{2} df^T \mathcal{H}(l) df + \nabla_f Z^T(l) df, \tag{14}$$

subject to

$$\mathcal{E}_r f(l) - b_r + \mathcal{E}_r df = 0, \tag{15}$$

where $b_r = (b_r^1, \ldots, b_r^k)$. Since $\mathcal{H}(l)$ is positive definite and $\mathcal{E}_r$ is of full rank, the constraint qualification of (9a)–(9b) is ensured. By the strong duality theory [2], solving (9a)–(9b) is equivalent to solving the dual problem (10). Thus, we have the following theorem:

Theorem 2. *Let df^*, λ^*, and $\hat{d}f(\lambda^*)$ be the optimal solutions of* $(9a)-(9b)$, (10), *and the minimization problem on the rhs of* (11) *with $\lambda = \lambda^*$, respectively; then $df^* = \hat{d}f(\lambda^*)$.*

3.2. Ascent Direction

Thus, we can use the proposed DPPQN method to solve (10) instead of solving (3a)–(3d). The DPPQN method uses the following iterations:

$$\lambda(t+1) = \lambda(t) + \beta(t)\Delta\lambda(t), \tag{16}$$

where t denotes the iteration index, $\beta(t) > 0$ is a step-size, and $\Delta\lambda(t)$ is obtained by solving the following linear equations:

$$[\nabla^2_{\lambda\lambda}\phi^u(\lambda(t)) - \delta I]\Delta\lambda(t) + \nabla\phi(\lambda(t)) = 0, \tag{17}$$

where $\nabla\phi(\lambda(t))$ is the gradient of $\phi(\lambda)$ with respect to λ at $\lambda(t)$, and $\phi^u(\lambda)$ is the unconstrained dual function defined by relaxing the inequality constraints on primal variables df from $\phi(\lambda)$ in (11), as shown below:

$$\begin{aligned}\phi^u(\lambda) = \min \sum_{k=1}^{K} &\left[\frac{1}{2} df^{k^T} H^k(l) df^k \right.\\ &\left. + \sum_{a=1}^{m} \frac{\partial Z_a(l)}{\partial f_a^k} df_a^k\right] \\ &+ \sum_{k=1}^{K} \lambda^{k^T}[E_r f^k(l) - b_r^k + E_r df^k].\end{aligned} \tag{18}$$

$\nabla^2_{\lambda\lambda}\phi^u(\lambda(t))$ is the Hessian matrix of $\phi^u(\lambda(t))$, δ is a positive real number, and I is an identity matrix. From [21, Chap. 13], we see that $\nabla_\lambda\phi(\lambda(t))$ and $\nabla^2_{\lambda\lambda}\phi^u(\lambda(t))$ can be computed by

$$\nabla_{\lambda^k}\phi(\lambda(t)) = E_r f^k(l) - b_r^k + E_r \hat{d}f^k(\lambda(t)), \quad k = 1, \ldots, K, \tag{19}$$

$$\nabla^2_{\lambda\lambda}\phi^u(\lambda(t)) = -\mathscr{E}_r \mathscr{H}^{-1}(l)\mathscr{E}_r^T. \tag{20}$$

Equation (19) can also be written as

$$\nabla_\lambda\phi(\lambda(t)) = \mathscr{E}_r f(l) - b_r + \mathscr{E}_r \hat{d}f(\lambda(t)). \tag{21}$$

The $\hat{d}f(\lambda(t))$ in (19) or in (21) is the optimal solution of the constraint-minimization problem on the rhs of (11), with $\lambda = \lambda(t)$. The value of $\hat{d}f(\lambda(t))$ should be obtained before solving (17) for $\Delta\lambda(t)$; the method for obtaining $\hat{d}f(\lambda(t))$ is presented in Section 3.5. Since $\mathscr{H}(l)$ is positive definite, and $\mathscr{E}_r$ has full rank, $\nabla^2_{\lambda\lambda}\phi^u(\lambda(t))$ as well as $\nabla^2_{\lambda\lambda}\phi(\lambda(t)) - \delta I$ are negative definite. Thus, the $\Delta\lambda(t)$ obtained from (17) is an ascent direction for (10).

As will be shown later in Section 3.6, if the step-size $\beta(t) \in (0, (2\delta/K'))$, where K' is the Lipschitz constant of the dual function $\phi(\lambda)$, then (16) is guaranteed to be an ascent method. Although we will prove the existence of and present a value for K' in Section 3.6, this K' gives us a very conservative step-size that will cause methodological inefficiency. Therefore, we next present a variable step-size rule similar to the step-size rule in Section 2 that relaxes the strong sufficient condition on the range of the step-size $\beta(t)$.

3.3. Determination of a Variable Step Size

Let $0 < \tau_D < 1$. The Armijo-type step-size rule for determining $\beta(t)$ is to increase m from 0 in increments of 1 and check whether the following inequality holds:

$$\begin{aligned}&\phi(\lambda(t) + \tau_D^m \Delta\lambda(t)) \\ &\qquad > \phi(\lambda(t)) + \frac{\delta}{2}\tau_D^m \|\Delta\lambda(t)\|_2^2.\end{aligned} \tag{22}$$

We set $\beta(t) = \tau_D^{m(t)}$, where $m(t)$ is the smallest nonnegative integer m for which (22) holds. From (22), we see that (16), being an ascent method, is ensured without requiring $\beta(t) \in (0, (2\delta/K'))$. [The existence of $m(t)$ is shown in Section 3.6, and further discussion of step-size is given in the comments following the proof.]

3.4. Advantages of Commodity Decomposition and Network Sparsity

3.4.1. Commodity Decomposition

Because $\mathscr{H}(l)$ as shown in (13) is a diagonal matrix, it can thus easily be observed from (20) and the structure of $\mathscr{E}_r$ in (12) that $\nabla^2_{\lambda\lambda}\phi^u(\lambda(t))$ as well as $\nabla^2_{\lambda\lambda}\phi^u(\lambda(t)) - \delta I$ are $(n-1)K \times (n-1)K$-block diagonal matrices, with each diagonal block submatrix corresponding to one commodity. Thus, the $(n-1)K \times (n-1)K$ linear equations (17) can be decomposed into K independent sets of $n - 1 \times n - 1$ linear equations, as shown in (23):

$$[\nabla^2_{\lambda\lambda}\phi^u(\lambda(t)) - \delta I]^k \Delta\lambda^k(t) + \nabla_{\lambda^k}\phi(\lambda(t)) = 0, \quad k = 1, \ldots, K, \tag{23}$$

where the $n - 1 \times n - 1$ matrix

$$[\nabla^2_{\lambda\lambda}\phi^u(\lambda(t)) - \delta I]^k = E_r H^k(l) E_r^T - \delta I^k \tag{24}$$

denotes the kth diagonal block submatrix of $\nabla^2_{\lambda\lambda}\phi^u(\lambda(t)) - \delta I$ corresponding to commodity k, and I^k is an $n - 1 \times n - 1$ identity matrix.

3.4.2. The Applicability of the Efficient Sparse-Matrix Technique

From (24), we see that only if node i and node ν are connected will there be a nonzero entry in the off-diagonal (i, ν)th position of $[\nabla^2_{\lambda\lambda}\phi^u(\lambda(t)) - \delta I]^k$. Consequently, the matrix $[\nabla^2_{\lambda\lambda}\phi^u(\lambda(t)) - \delta I]^k$ is as sparse as the network. Hence, for each commodity k, we may use an efficient sparse-matrix technique to solve the linear equation (23) once the optimal solution $\hat{d}f(\lambda(t))$ for the constraint-minimization on the rhs of (11) is obtained.

Remark 1. *The K-independent sets of linear equations (23) can be solved in parallel if parallel processors are available.*

Remark 2. *In the solution process of the DPPQN method, $[\nabla^2_{\lambda\lambda}\phi^u(\lambda(t)) - \delta I]^k$ is a constant matrix for all k.*

3.5. The Two-phase Method for Solving $\hat{d}f(\lambda(t))$ and the Successive Projection and (Truncated) Seal Algorithm

3.5.1. The Two-phase Method

To calculate the $\nabla_\lambda\phi(\lambda(t))$ needed in (23) to solve for $\Delta\lambda(t)$, we have to solve the problem of constraint-minimization on the rhs of (11) to obtain the value of the vector $\hat{d}f(\lambda(t))$.

$\sum_{k=1}^{K} \lambda^{k^T} E_r df^k$ can be written as $\sum_{a=1}^{m} [\lambda_{a\otimes} - \lambda_{a\odot}]^T df_a$, where $(a\otimes, a\odot)$ denotes the directed arc a from node $a\otimes$ to node $a\odot$, $\lambda_{a\otimes} = (\lambda^1_{a\otimes}, \ldots, \lambda^K_{a\otimes})$, and $\lambda_{a\odot} = (\lambda^1_{a\odot}, \ldots, \lambda^K_{a\odot})$; furthermore, $\lambda_{a\odot} = (0, \ldots, 0)$ if $a\odot = n$. Thus, we can rewrite (11) as

$$\begin{aligned}\phi(\lambda) = \min_{f(l)+df\in\mathcal{F}} \sum_{a=1}^{m} [\tfrac{1}{2} df_a^T H_a(l) df_a \\ + \nabla_{f_a} Z_a^T(l) df_a] + \sum_{a=1}^{m} [\lambda_{a\otimes} - \lambda_{a\odot}]^T df_a \\ + \sum_{k=1}^{K} \lambda_k^T [E_r f^k(l) - b_r^k].\end{aligned} \tag{25}$$

Thus, the problem of constraint-minimization on the rhs of (25) is equivalent to that on the rhs of (11).

Since the value of $\partial^2 Z_a(l)/\partial f_a^{k^2}$ is the same for each k, we define

$$\gamma_a = \begin{cases} \dfrac{\partial^2 Z_a(l)}{\partial f_a^{k^2}} & \text{if } \dfrac{\partial^2 Z_a(l)}{\partial f_a^{k^2}} \geq \gamma, \\ \gamma & \text{otherwise.} \end{cases} \tag{26}$$

Then, from (4), the $K \times K$ diagonal matrix

$$H_a(l) = \gamma_a I. \tag{27}$$

Thus, the problem of constraint-minimization on the rhs of (25) can be viewed as a projection problem that can be solved in two phases, as described in the following lemma:

Lemma 3. *The solution, $\hat{d}f(\lambda(t))$, for the problem of constraint-minimization on the rhs of (25), with $\lambda = \lambda(t)$, can be solved in two phases.*

Phase 1: Calculate

$$\tilde{d}f_a(\lambda(t)) = -\frac{\dfrac{\partial Z_a(l)}{\partial f_a} + \lambda_{a\otimes}(t) - \lambda_{a\odot}(t)}{\gamma_a}, \quad \text{for every } a. \tag{28}$$

Phase 2: Project $\tilde{d}f_a(\lambda(t))$ onto the set $\mathcal{F}_a - f_a(l)$ for every arc a; the projection is $\hat{d}f(\lambda(t))$.

Proof. The problem of constraint-minimization on the rhs of (25), with $\lambda = \lambda(t)$, is equivalent to the following problem:

$$\begin{aligned}\min_{f(l)+df\in\mathcal{F}} \sum_{a=1}^{m} [\tfrac{1}{2} df_a^T H_a(l) df_a + \nabla_{f_a} Z^T(l) df_a \\ + (\lambda_{a\otimes}(t) - \lambda_{a\odot}(t))^T df_a],\end{aligned} \tag{29}$$

because $\lambda(t)$, $f(l)$ and b_r are constants. From the definition of $\mathcal{F}$, we see that $\mathcal{F} - f(l) = \cup_{a=1}^{m}(\mathcal{F}_a - f_a(l))$, and $(\mathcal{F}_{a'} - f_{a'}(l)) \cap (\mathcal{F}_{a''} - f_{a''}(l) = \emptyset$ if $a' \neq a''$; then, also by (27), we can rewrite (29) as

$$\begin{aligned}\sum_{a=1}^{m} \min_{f_a(l)+df_a\in\mathcal{F}_a} [\tfrac{1}{2}\gamma_a df_a^T df_a + (\nabla_{f_a} Z_a(l) \\ + \lambda_{a\otimes}(t) - \lambda_{a\odot}(t))^T df_a],\end{aligned} \tag{30}$$

which is separable and can be solved independently by solving the m subproblems with each subproblem in the form

$$\begin{aligned}\min_{f_a(l)+df_a\in\mathcal{F}_a} [\tfrac{1}{2}\gamma_a df_a^T df_a + (\nabla_{f_a} Z_a(l) \\ + \lambda_{a\otimes}(t) - \lambda_{a\odot}(t))^T df_a].\end{aligned} \tag{31}$$

Since γ_a and $\nabla_{f_a} Z_a(l) + \lambda_{a\otimes}(t) - \lambda_{a\odot}(t)$ are constants in (31), the solution of (31) is equivalent to the solution of the following problem:

$$\min_{f_a(l)+df_a \in \mathcal{F}_a} \Bigg[\frac{1}{2} \gamma_a df_a^T df_a + (\nabla_{f_a} Z_a(l) + \lambda_{a\otimes}(t) - \lambda_{a\odot}(t))^T df_a + \frac{1}{2\gamma_a} (\nabla_{f_a} Z_a(l) + \lambda_{a\otimes}(t) - \lambda_{a\odot}(t))^T (\nabla_{f_a} Z_a(l) + \lambda_{a\otimes}(t) - \lambda_{a\odot}(t)) \Bigg]. \tag{32}$$

It can easily be observed that the solution of (32) is equivalent to the solution of the following projection problem:

$$\min_{f_a(l)+df_a \in \mathcal{F}_a} \left\| df_a + \frac{1}{\gamma_a} [\nabla_{f_a} Z_a(l) + \lambda_{a\otimes}(t) - \lambda_{a\odot}(t)] \right\|_2^2. \tag{33}$$

The solution of (33) is $\hat{d}f_a(\lambda(t))$, which is the projection of the following vector:

$$\tilde{d}f_a(\lambda(t)) = -\frac{1}{\gamma_a} [\nabla_{f_a} Z_a(l) + \lambda_{a\otimes}(t) - \lambda_{a\odot}(t)]$$

onto the set $\mathcal{F}_a - f_a(l)$. This completes the proof. ■

The above lemma motivated us to develop an efficient projection algorithm to obtain the projection $\hat{d}f(\lambda(t))$ in Phase 2. As we can see in the proof, an arc decomposition—the second decomposition effect—is achieved in this two-phase method, which also contributes to the computational efficiency of our method.

3.5.2. Successive Projection and (Truncated) Seal Algorithm

Han pointed out in [16] that for projection onto the intersection of closed convex sets formed from linear inequality constraints cases exist in which straightforward successive projection may fail; for this reason, he proposed a successive projection method to solve the general projection problem. He modified the straightforward successive projection method by adding to the projected point a calculated outward normal vector before each projection. He showed that the sequence of normal vectors will converge to the solution of the dual projection problem. His method is an infinite iterative process; however, the constraint set $\mathcal{F}_a - f_a(l)$ is a case to which the straightforward successive projection method can apply, but the disadvantage is it also takes infinite iterations. For example, Figure 1 shows the process of a straightforward successive projection for projecting a point P onto the constraint set $\{(df_a^1, df_a^2) \mid f_a^k(l) + df_a^k \geq 0,\ k = 1, 2,\ f_a^1(l) + df_a^1 + f_a^2(l) + df_a^2 \leq d_a\}$, which is $\mathcal{F}_a - f_a(l)$ for $K = 2$ denoted by the shaded area in Figure 1; this process takes infinite iterations to obtain the solution point $(-f_a^1, d_a - f_a^2(l))$, as shown in Figure 1. To circumvent this disadvantage, we propose a K-iteration successive projection and (truncated) seal algorithm to obtain the $\hat{d}f(\lambda(t))$.

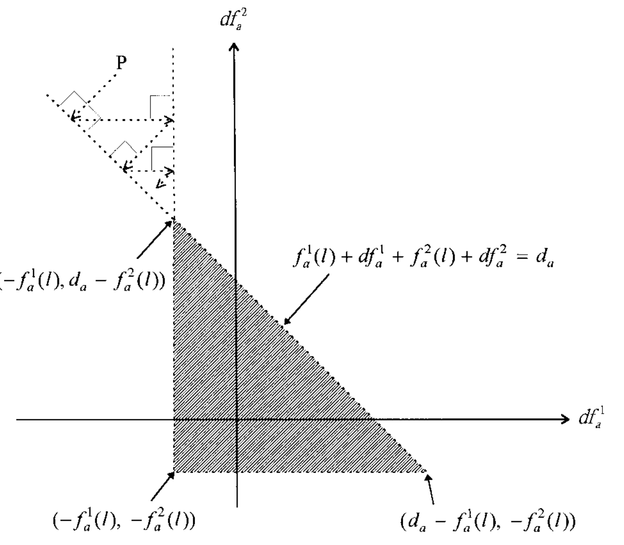
df_a^2
P
$f_a^1(l) + df_a^1 + f_a^2(l) + df_a^2 = d_a$
$(-f_a^1(l), d_a - f_a^2(l))$
df_a^1
$(-f_a^1(l), -f_a^2(l))$
$(d_a - f_a^1(l), -f_a^2(l))$

Fig. 1. Successive straightforward projection example.

Successive Projection and Truncated (*Seal*) *Algorithm*

Notation: The set $\mathcal{K} = \{1, 2, \ldots, K\}$; $|(\cdot)|$ denotes the cardinality of the set $(\cdot)$.

Step SP0. Execute the following for every arc a.

Step SP1. Set $\mathcal{K}(a) = \mathcal{K}$ and repeat the following for every k: if $f_a^k(l) + \tilde{d}f_a^k(\lambda(t)) < 0$, set $\tilde{d}f_a^k(\lambda(t)) = -f_a^k(l)$ and $\mathcal{K}'(a) = \mathcal{K}'(a) \backslash \{k\}$.

Step SP2. If $\Sigma_{k \in \mathcal{K}'(a)} (f_a^k(l) + \tilde{d}f_a^k(\lambda(t))) > d_a$, then calculate

$$\eta = \frac{d_a - \sum_{k \in \mathcal{K}'(a)} (f_a^k(l) + \tilde{d}f_a^k(\lambda(t)))}{|\mathcal{K}'(a)|},$$

and go to Step SP3; otherwise, set $\hat{d}f_a^k(\lambda(t)) = \tilde{d}f_a^k(\lambda(t))$, for every $k \in \mathcal{K}$, and stop.

Step SP3. Update $\tilde{d}f_a^k(\lambda(t)) = \tilde{d}f_a^k(\lambda(t)) + \eta$, for every $k \in \mathcal{K}'(a)$.

Step SP4. Calculate $\tau = \min_{k \in \mathcal{K}'(a)}(f_a^k(l) + \tilde{d}f_a^k(\lambda(t)))$. If $\tau \geq 0$, set $\hat{d}f_a^k(\lambda(t)) = \tilde{d}f_a^k(\lambda(t))$, for every

$k \in \mathcal{K}$, and stop; otherwise, set $\tilde{d}f_a^k(\lambda(t)) := \tilde{d}f_a^k(\lambda(t)) - \tau$, for every $k \in \mathcal{K}'(a)$. Then, repeat the following for every $k \in \mathcal{K}'(a)$: If $f_a^k(l) + \tilde{d}f_a^k(\lambda(t)) = 0$, set $\tilde{d}f_a^k(\lambda(t)) = -f_a^k(l)$ and $\mathcal{K}'(a) = \mathcal{K}'(a)\setminus\{k\}$. Return to Step SP2.

Our successive projection and (truncated) seal algorithm can be illustrated as follows. Let $\mathcal{F}'_a = \{(df_a^1, \ldots, df_a^K) | 0 \le f_a^k(l) + df_a^k$, for every $k\}$ and $\mathcal{F}''_a = \{(df_a^1, \ldots, df_a^K) | \Sigma_{k=1}^K f_a^k(l) + df_a^k = d_a\}$. If $\tilde{d}f_a(\lambda(t))$ is not in $\mathcal{F}'_a$, then in Step SP1, we project it onto $\mathcal{F}'_a$; otherwise, we skip this step. However, it should be noted that in Step SP1 if any component $\tilde{d}f_a^k(\lambda(t))$ is ever set equal to $-f_a^k(l)$ it will be kept fixed at that value for the rest of the process; this action is the *seal* step. In other words, when the component $\tilde{d}f_a^k(\lambda(t))$ is projected to the corresponding boundary $f_a^k(l) + \tilde{d}f_a^k(\lambda(t)) = 0$, it will stay at that boundary for the rest of the process. After Step SP1, if the solution has not yet been obtained, we project the resulting point onto $\mathcal{F}''_a$, as described by Steps SP2 and SP3. If the resulting point obtained after Step SP3 belongs to $\mathcal{F}'_a$, we stop; otherwise, we pull that point along the opposite projection direction obtained in Step SP2 back to the boundary of $\mathcal{F}'_a$, as shown in Step SP4 and check which component $\tilde{d}f_a^k(\lambda(t))$ lying on the corresponding boundary, $f_a^k(l) + \tilde{d}f_a^k(\lambda(t)) = 0$, and that component will be fixed at the value $-f_a^k(l)$—this action is the *truncated seal* step. Execution then returns to Step SP2 to continue the process.

Advantages of the Algorithm: (i) Because of the seal and truncated seal steps, our algorithm converges within K iterations. This can easily be observed since in each iteration at least one component is sealed and there are only K components in $\tilde{d}f(\lambda(t))$. (ii) Because the projection procedures for different arcs are independent, our algorithm is suitable for parallel processing if parallel processors are available.

Convergence of the Algorithm: That K-iteration successive projection and (truncated) seal algorithm obtains the exact projection in at most K iterations is stated in the following theorem, while the proof, which is very complicated, is given in the Appendix.

Theorem 3. *For each arc a, the successive projection and (truncated) seal algorithm will stop at the solution point of projecting $\tilde{d}f_a(\lambda(t))$ onto the set $\mathcal{F}_a - f_a(l)$ in at most K iterations.*

3.6. Proof of the Convergence of the Dual Projected Pseudo-Quasi-Newton Method

We can summarize the DPPQN method in the following: For a given $\lambda(t)$, using (28) and the successive projection and (truncated) seal algorithm, we can obtain the solution $\hat{d}f(\lambda(t))$ of the constraint-minimization problem on the rhs of (11). We can then proceed to set up the K-independent sets of linear equations (23) that will be solved independently for each k using a sparse-matrix technique to obtain $\Delta\lambda(t)$. After that, we determine the variable step-size $\beta(t)$ according to (22) and update $\lambda(t+1)$ using (16) and then proceed with the next iteration of the DPPQN method. In this section, we prove the convergence of this iterative method for solving the dual problem (10).

Because $\Delta\lambda(t)$ is an ascent direction as indicated previously in Section 3.2, and the objective function $\phi(\lambda)$ is concave and bounded from above, we can modify the convergence theorem in [5, Proposition 2.3] to obtain the following lemma:

Lemma 4. *Suppose that there exists a positive constant K' such that*

$$\|\nabla_\lambda\phi(\lambda') - \nabla_\lambda\phi(\lambda'')\|_2 \le K'\|\lambda' - \lambda''\|_2, \quad \forall\lambda', \lambda'' \in \Re^{(n-1)K}, \tag{34}$$

then (a)

$$\phi(\lambda(t) + \beta(t)\Delta\lambda(t)) \ge \phi(\lambda(t)) + \beta^2(t)\left(\frac{\delta}{\beta(t)} - \frac{K'}{2}\right)\|\Delta\lambda(t)\|_2^2;$$

(b) if $0 < \beta(t) < \frac{2\delta}{K'}$, then (16) is an ascent method; and (c) if $0 < \beta(t) < \frac{2\delta}{K'}$, for every t, then any limit point λ^* generated by (16) satisfies $\lim_{t\to\infty} \nabla_\lambda\phi(\lambda(t)) = 0$ and maximizes $\phi(\lambda)$.

However, we need to show the existence of the Lipschitz constant K' required in the assumption of Lemma 4 as follows:

Lemma 5. *There exists a positive constant K' such that $\|\nabla_\lambda\phi(\lambda') - \nabla_\lambda\phi(\lambda'')\|_2 \le K'\|\lambda' - \lambda''\|_2$ for every $\lambda', \lambda'' \in \Re^{(n-1)K}$.*

Proof. From (21),

$$\begin{aligned}\nabla_\lambda\phi(\lambda') &= \mathcal{E}_r f(l) - b_r + \mathcal{E}_r\hat{d}f(\lambda')\\ \nabla_\lambda\phi(\lambda'') &= \mathcal{E}_r f(l) - b_r + \mathcal{E}\hat{d}f(\lambda''),\end{aligned} \tag{35}$$

where $\hat{d}f(\lambda')$ and $\hat{d}f(\lambda'')$ are solutions of the r.h.s. constraint-minimization problem of (11) for $\lambda = \lambda'$ and $\lambda = \lambda''$, respectively.

From (35), we have

$$\begin{aligned}\|\nabla_\lambda\phi(\lambda') - \nabla_\lambda\phi(\lambda'')\|_2 &= \|\mathscr{E}_r\hat{d}f(\lambda') - \mathscr{E}_r\hat{d}f(\lambda'')\|_2 \\ &\le \|\mathscr{E}_r\|_2\|\hat{d}f(\lambda') - \hat{d}f(\lambda'')\|_2.\end{aligned} \tag{36}$$

Since $\hat{d}f(\lambda)$ is the projection of $\tilde{d}f(\lambda)$ onto the set $\mathscr{F} - f(l)$, then by the projection theory in [5, p. 211],

$$\|\hat{d}f(\lambda') - \hat{d}f(\lambda'')\|_2 \le \|\tilde{d}f(\lambda') - \tilde{d}f(\lambda'')\|_2. \tag{37}$$

From (26), we see that $\gamma \le \gamma_a$ for every a. Then, from (28), we obtain

$$\begin{aligned}\|\tilde{d}f(\lambda') - \tilde{d}f(\lambda'')\|_2 &\le \frac{1}{\gamma}\|\mathscr{E}_r^T\lambda' - \mathscr{E}_r^T\lambda''\|_2 \\ &\le \frac{1}{\gamma}\|\mathscr{E}_r^T\|_2\|\lambda' - \lambda''\|_2.\end{aligned} \tag{38}$$

From (36), (37), and (38), we obtain

$$\|\nabla_\lambda\phi(\lambda') - \nabla_\lambda\phi(\lambda'')\|_2 \le \frac{1}{\gamma}\|\mathscr{E}_r\|_2\|\mathscr{E}_r^T\|_2\|\lambda' - \lambda''\|_2.$$

Let

$$K' = \frac{1}{\gamma}\|\mathscr{E}_r\|_2\|\mathscr{E}_r^T\|_2, \tag{39}$$

and the proof is complete. ■

Because the value of K' is large, as shown in (39), as previously printed out in Section 3.3, the conservative range $(0, (2\delta/K'))$ for the step-size $\beta(t)$ provided in Lemma 4 makes (16) inefficient. To cope with this inefficiency, we presented in Section 3.3 an Armijo-type step-size determination rule, in which the existence of a nonnegative integer m for which the inequality (22) holds should be verified; this justification is given below.

Lemma 6. *There exists a nonnegative integer m such that inequality (22) holds.*

Proof. From (a) of Lemma 4,

$$\begin{aligned}&\phi(\lambda(t) + \beta(t)\Delta\lambda(t)) \\ &\quad > \phi(\lambda(t)) + \beta^2(t)\left(\frac{\delta}{\beta(t)} - \frac{K'}{2}\right)\|\Delta\lambda(t)\|_2^2,\end{aligned} \tag{40}$$

from which we can further derive that if $0 < \beta(t) < (\delta/K')$,

$$\begin{aligned}&\phi(\lambda(t) + \beta(t)\Delta\lambda(t)) \\ &\quad > \phi(\lambda(t)) + \frac{\delta}{2}\beta(t)\|\Delta\lambda(t)\|_2^2.\end{aligned} \tag{41}$$

Because $0 < \tau_D < 1$, $\{\tau_D^m\}$ is a monotonically decreasing sequence; there must exist a nonnegative integer m such that $0 < \tau_D^m < (\delta/K')$. Then, using (41), we prove the existence of m for which (22) holds. ■

Comment 1. *A first glance at the proof shows that the Armijo-type step-size does not seem to improve upon the magnitude of the step-size provided in Lemma 4. In fact, the smallest $m(t)$ for which (22) holds is very likely to have $\tau_D^{m(t)} \gg (\delta/K')$, because the inequality $0 < \beta(t) < (\delta/K')$ is a sufficient condition for (41) to hold.*

From Lemmas 4, 5, and 6 and Theorem 3, the following theorem for the convergence of the DPPQN method can be easily proven:

Theorem 4. *Any limit point λ^* of the sequence $\{\lambda(t)\}$ generated by (16) with a step-size determined according to (22) satisfies $\nabla_\lambda\phi(\lambda) = 0$ and maximizes $\phi(\lambda)$.*

4. THE COMPLETE ALGORITHM FOR NMNFPs

Theorems 1 and 4 together show the convergence of the projected Jacobi method combined with the DPPQN method. We now summarize the overall method for solving (1a)–(1d) in the following algorithm steps:

STEP 1. Set values for the parameters $\delta > 0$, τ_J, τ_D, $(0 < \tau_J, \tau_D < 1)$; set $l = 1$ and initially guess the value of f, which can be infeasible, and set $f(l) = f$.

STEP 2. If $l = 1$ and the value of λ has not yet been assigned, guess the value of λ based on some rule; otherwise, set λ at its previous value. Set $t = 1$.

STEP 3. Compute $\tilde{d}f(\lambda(t))$ using (28).

STEP 4. Use the successive projection and (truncated) seal algorithm to obtain the projection $\hat{d}f(\lambda(t))$ by projecting $\tilde{d}f(\lambda(t))$ onto the set $\mathscr{F} - f(l)$.

STEP 5. Compute $\nabla_{\lambda^k}\phi(\lambda(t))$, $[\nabla^2_{\lambda\lambda}\phi^u(\lambda(t)) - \delta I]^k$, for every k by (19) and (24), respectively.

STEP 6. Use a sparse-matrix technique to solve (23) for every k individually and obtain $\Delta\lambda(t)$.

STEP 7. Determine a value for $\beta(t) = \tau_D^{m(t)}$, where $m(t)$ is the smallest nonnegative integer such that (22) holds, and update $\lambda(t + 1)$ by (16).

STEP 8. Check whether $\|\beta(t)\Delta\lambda(t)\| \le \varepsilon_1$ to test the

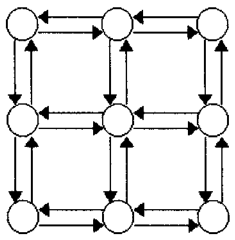

Fig. 2(a).

convergence of (16); if yes, set $df^*(l) = \hat{d}f(\lambda(t))$ and go to Step 9; otherwise, set $t = t + 1$ and return to Step 3.

Step 9. If $l = 1$, update $f(l+1) = f(l) + df^*(l)$; otherwise, determine $\alpha(l) = \tau_J^{m(l)}$, where $m(l)$ is the smallest nonnegative integer for which (8) holds, and update $f(l+1)$ using (2).

Step 10. Check whether $\|\alpha(l)df^*(l)\| \le \varepsilon_2$ to test the convergence of (2); if yes, stop; otherwise, go to Step 2.

Remark 3. *Because the initially guessed $f(1)$ may be infeasible, we have to set $\alpha(1) = 1$ in Step 9 in the first iteration of the projected Jacobi method so as to obtain a feasible $f(2)$. Consequently, the condition required in Theorem 1 that the sequence $\{f(l)\}$ start from a feasible f is satisfied.*

5. SIMULATION EXAMPLES

5.1. A Programming Consideration

Theoretically, it takes an infinite number of iterations to achieve infinite accuracies in the convergence check of Steps 8 and 10. In general, a good finite accuracy is sufficient for practical applications.

We observed from extensive simulated examples that it takes many iterations of the projected Jacobi method to push the accuracy from $\varepsilon_2 = 10^{-3}$ to $\varepsilon_2 = 10^{-4}$ in Step 10. This is because the convergence rate of the projected Jacobi method is linear, not quadratic. However, there are only negligible deviations between the objective values obtained using $\varepsilon_2 = 10^{-3}$ and the exact optimal objective values as verified by commercial optimization tools for numerous examples. Therefore, we normally select the accuracies $\varepsilon_1 = \varepsilon_2 = 10^{-3}$ for our algorithm. However, we apply a postprocessing step to refine the final solution that we obtained to enhance the degree of feasibility. The postprocessing step executes the inner loop, which ranges from Step 2 to Step 8, of our complete algorithm one more time after the final solution has been obtained, but this time using $\varepsilon_1 = 10^{-6}$ in $\|\beta(t)\Delta\lambda(t)\|_\infty < \varepsilon_1$ for the convergence check in Step 8. The magnitude of $df^*(l)$ obtained from the postprocessing step should be negligible; however, when we add this $df^*(l)$ to the final solution, the flow-balance equation will be satisfied up to the sixth significant decimal place.

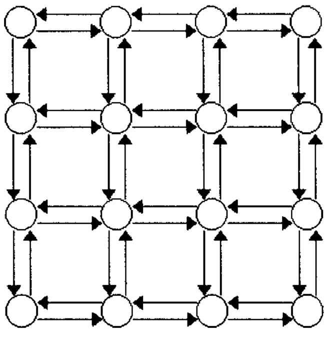

Fig. 2(b).

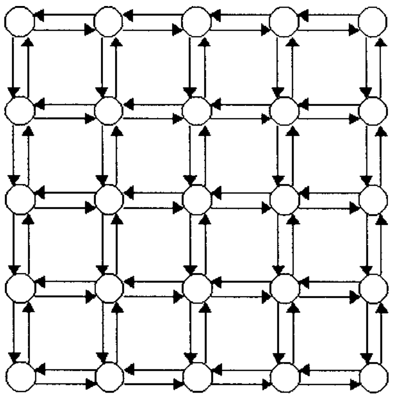

Fig. 2(c).

5.2. Examples

Since our method fully exploits the structural advantages of the NMNFP in the aspects of commodity decomposi-

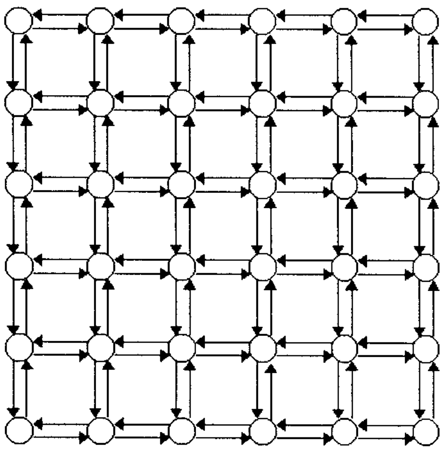

Fig. 2(d).

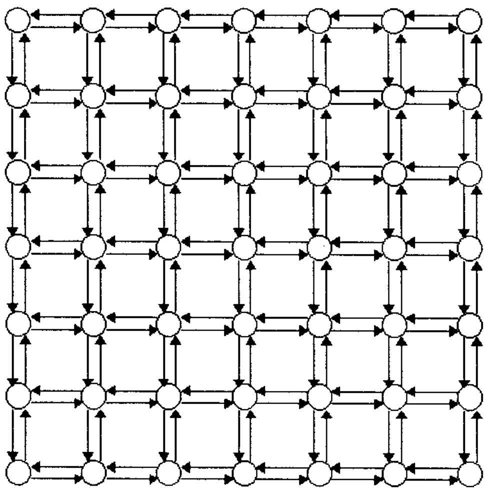
Fig. 2(e).

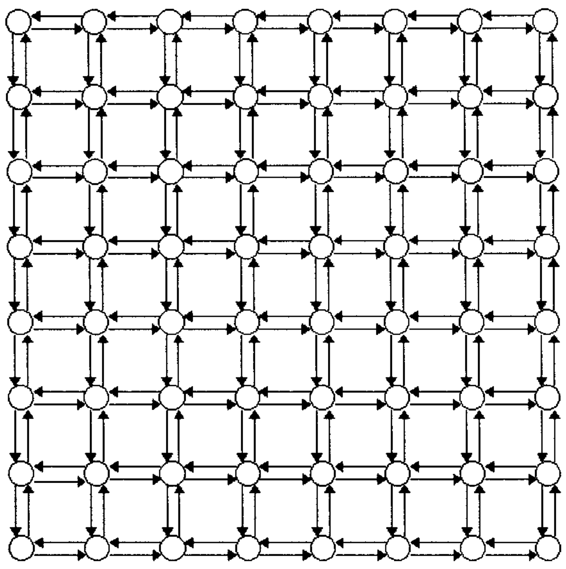
Fig. 2(f).

tion, arc decomposition, special types of coupling capacity constraints, and network sparsity, as described in Section 3, we have chosen example networks of various sizes and have simulated cases involving different commodities in each network so as to test the computational efficiency of our proposed algorithm, as well as to compare it with typical network-based methods like Frank–Wolfe and Frank–Wolfe with PARTAN algorithms for NMNFP.

We selected six grid-type networks, as shown in Figures 2(a)–(f), with varying numbers of nodes and arcs for testing and comparing our algorithm. In each network, we tested five cases from two commodities to six commodities. We set the capacity of each arc, d_a, at 30 and assumed the flow value of each commodity sent from source node to destination, F_i^k, to be 10. We considered a quadratic cost function of the following form: $\frac{1}{2}\sum_{a=1}^{m}(\sum_{k=1}^{K} f_a^k)^2$.

With the following setup: $\varepsilon_1 = \varepsilon_2 = 10^{-3}$, $\tau_J = 0.8$, $\tau_D = 0.8$, initial zero flow, and a postprocessing step in which $\varepsilon_1 = 10^{-6}$, we used our method to solve the above 30 NMNFPs on a Sparc-10 workstation. The consumption of CPU time and the final objective value obtained by our method for each case on each network are reported in Tables I–VI. Note that the CPU time was clocked for the complete algorithm listed in Section 4.

To demonstrate the computational efficiency of our method and verify our solution, we used the Frank–Wolfe algorithm and Frank–Wolfe with PARTAN algorithm to solve the same 30 NMNFPs on the same Sparc-10 workstation.

One point we would like to emphasize is the use of the price-directive decomposition method to solve the linearized multicommodity flow problems induced in the Frank–Wolfe and Frank–Wolfe with PARTAN algorithms. The price-directive decomposition method is one of the most efficient methods for dealing with linear MNFPs with coupling capacity constraints.

The consumption of CPU time and the final objective

TABLE I. Comparison of our method with Frank–Wolfe and Frank–Wolfe with PARTAN algorithms for solving the network NMNFP shown in Figure 2(a)

	Final objective value			CPU time (seconds)			Speed-up ratio	
No. commodities	Our method	F–W algorithm	F–W PARTAN algorithm	Our method (I)	F–W algorithm (II)	F–W PARTAN algorithm (III)	II/I	III/I
2	159.7	160.2	159.7	0.06	1.90	0.85	31.67	14.17
3	297.6	298.2	297.9	0.09	9.95	1.86	110.56	20.67
4	449.3	450.0	449.5	0.17	25.30	4.36	148.82	25.65
5	560.1	560.5	560.3	0.12	42.69	4.62	355.75	38.50
6	662.6	663.3	662.8	0.19	97.52	9.19	513.26	48.37

TABLE II. Comparison of our method with Frank–Wolfe and Frank–Wolfe with PARTAN algorithms for solving the network NMNFP shown in Figure 2(b)

	Final objective value			CPU time (seconds)			Speed-up ratio	
No. commodities	Our method	F–W algorithm	F–W PARTAN algorithm	Our method (I)	F–W algorithm (II)	F–W PARTAN algorithm (III)	II/I	III/I
2	134.6	135.2	135.0	0.1	23.18	23.18	231.8	125.40
3	220.2	220.8	202.7	0.15	60.37	29.46	402.47	196.40
4	298.4	299.0	298.6	0.23	133.23	36.23	579.26	157.50
5	387.7	388.4	388.0	0.51	308.06	83.25	604.04	163.64
6	469.7	470.4	469.9	0.33	615.97	92.82	1866.58	281.27

values obtained by the Frank–Wolfe and Frank–Wolfe with PARTAN algorithms for each case on each network are also reported in Tables I–VI. However, for the cases involving five and six commodities on the network shown in Figure 2(f), the Frank–Wolfe and Frank–Wolfe with PARTAN algorithms have numerical difficulties in getting the solution for such large-dimensioned NMNFPs.

From Tables I–VI, we see that the Frank–Wolfe with PARTAN algorithm is better than the Frank–Wolfe algorithm with respect to computational efficiency as we expected. From the final objective value columns of Tables I–VI, we can see that our method obtained better objective-value solutions than did either the Frank–Wolfe or Frank–Wolfe with PARTAN algorithm in all cases. Furthermore, the computational efficiency of our method can also be easily observed in terms of the CPU time consumption. For example, in the case involving four commodities, the speed-up ratio of our complete algorithm compared with the Frank–Wolfe with PARTAN algorithm increase dramatically from 25 in the nine-node network to 4000 in the 64-node network. For the 36-node network [Fig. 2(d)], the speed-up ratio of our method compared with the Frank–Wolfe with PARTAN algorithm increased dramatically from 900 in the two-commodity case to 4000 in the six-commodity case. These observations strongly support the claim that our method is very efficient for large-dimensioned NMNFPs with respect to network size and number of commodities.

Comment 2. (*a*) *The setups for testing the CPU times of our method, the Frank–Wolfe algorithm, and the Frank–Wolfe with PARTAN algorithm were consistent and excluded the I/O times in all computer runs.* (*b*) *The CPU times for our complete algorithm reported in Tables I–VI include the CPU times for the sparse-matrix linear equation solver and successive projection and* (*truncated*) *seal algorithm subroutines. However, the CPU time for executing the sparse-matrix linear equation solver alone was too brief to be recorded in the workstation. The same situation occurred with the successive projection and* (*truncated*) *seal algorithm.*

Comment 3. *Concerning the striking speed-up ratio of our method compared with the Frank–Wolfe with PARTAN algorithm, we have the following observations:* (*a*) *The Frank–Wolfe with PARTAN algorithm is a gradient-*

TABLE III. Comparison of our method with Frank–Wolfe and Frank–Wolfe with PARTAN algorithms for solving the network NMNFP shown in Figure 2(c)

	Final objective value			CPU time (seconds)			Speed-up ratio	
No. commodities	Our method	F–W algorithm	F–W PARTAN algorithm	Our method (I)	F–W algorithm (II)	F–W PARTAN algorithm (III)	II/I	III/I
2	120.5	121.1	121.0	0.33	103.74	49.47	314.36	149.91
3	183.1	183.8	183.6	0.35	217.87	162.06	622.49	463.03
4	250.9	251.5	251.4	0.42	477.04	303.22	1135.81	721.95
5	420.1	420.7	420.5	0.49	1708.04	857.69	3485.80	1750.39
6	510.9	511.5	511.3	0.74	3070.54	1526.26	4149.38	2062.51

TABLE IV. Comparison of our method with Frank–Wolfe and Frank–Wolfe with PARTAN algorithms for solving the network NMNFP shown in Figure 2(d)

No. commodities	Final objective value			CPU time (seconds)			Speed-up ratio	
	Our method	F–W algorithm	F–W PARTAN algorithm	Our method (I)	F–W algorithm (II)	F–W PARTAN algorithm (III)	II/I	III/I
2	174.4	175.0	174.9	0.23	258.67	213.97	1124.65	930.30
3	263.3	263.9	263.8	0.51	981.85	726.01	1925.20	1423.55
4	318.4	319.0	318.8	0.87	2213.39	1664.85	2544.13	1913.62
5	381.3	381.9	381.8	0.91	4383.18	2982.57	4816.68	3277.55
6	471.9	472.5	472.4	1.27	8359.68	5067.42	6582.43	3990.09

type feasible direction method; it suffers from the slow convergence rate especially when the dimension of the problem is large. Furthermore, the coupling capacity constraints may severely restrict the Frank–Wolfe algorithm in obtaining a reasonable step-size; this factor makes the Frank–Wolfe algorithm even more numerically stiff. It may be this point as to why researchers usually used the Frank–Wolfe algorithm to deal with NMNFPs without coupling capacity constraints [10, 17]. *In contrast to the Frank–Wolfe algorithm, our method is a quasi-Newton-type method, we developed efficient methods to deal with the coupling-capacity constraints, and we employ the sparse-matrix technique to solve for the ascent direction. We successfully applied decomposition techniques to reduce the complications caused by large dimensions. Because our method has so many advantages and the conventional methods such as the Frank–Wolfe algorithm have difficulties in dealing with coupling capacity constraints, our method results in a dramatic performance vs. the Frank–Wolfe with PARTAN algorithm.* (*b*) *To further verify the advantages of our method, we also solve NMNFPs by the quadratic programming subroutine in the commercial optimization software package IMSL. The IMSL quadratic programming subroutine requires a large computer memory. When the size of the NMNFP on a grid network exceeds* 36 *nodes and four commodities, the memory space of our workstation is insufficient to execute the IMSL subroutine. The IMSL quadratic programming subroutine uses a sophisticated technique to deal with the coupling capacity constraints; however, it does not have the structural advantages of NMNFP. Therefore, for the* 36-*node network, the speed-up ratio of our algorithm compared with the IMSL subroutine is about* 200 *in the four-commodity case.*

Comment 4. *The existing dual-type methods* [4, 22, 25] *do not explore the structural advantages of NMNFPs with coupling capacity constraints as we have achieved in the proposed method. Thus, the efficiency of our method should not be attributed to the dual approach.*

6. CONCLUSION

We have developed a method that combines the well-known projected Jacobi method with a new dual projected

TABLE V. Comparison of our method with Frank–Wolfe and Frank–Wolfe with PARTAN algorithms for solving the network NMNFP shown in Figure 2(e)

No. commodities	Final objective value			CPU time (seconds)			Speed-up ratio	
	Our method	F–W algorithm	F–W PARTAN algorithm	Our method (I)	F–W algorithm (II)	F–W PARTAN algorithm (III)	II/I	III/I
2	148.0	148.5	148.5	0.64	764.44	571.74	1194.44	893.34
3	199.1	199.7	199.6	1.47	2031.72	1657.92	1382.12	1127.84
4	282.3	282.9	282.8	1.34	5303.73	3993.94	3958.01	2980.55
5	351.5	352.1	351.9	1.19	10213.32	7400.61	8582.62	6219
6	421.8	422.4	422.2	1.73	17838.4	13176.9	10311.2	7616.7

TABLE VI. Comparison of our method with Frank–Wolfe and Frank–Wolfe with PARTAN algorithms for solving the network NMNFP shown in Figure 2(f)

No. commodities	Final objective value: Our method	Final objective value: F–W algorithm	Final objective value: F–W PARTAN algorithm	CPU time (seconds): Our method (I)	CPU time (seconds): F–W algorithm (II)	CPU time (seconds): F–W PARTAN algorithm (III)	Speed-up ratio: II/I	Speed-up ratio: III/I
2	177.2	177.8	177.6	1.13	1938.02	1658.37	1715.06	1467.58
3	235.8	242.8	242.7	2.18	5409.83	4245.44	2481.57	1947.45
4	306.7	307.3	307.2	2.31	11735.93	9211.74	5080.49	3987.77
5	391.7	—	—	2.49	—	—	—	—
6	497.0	—	—	4.92	—	—	—	—

pseudo-quasi-Newton method for solving NMNFPs with convex objective functions.

We have experienced a dramatic speed-up ratio compared with the Frank–Wolfe with PARTAN algorithm associated with the price-directive decomposition method. Although it may not be the best network-based code for solving NMNFPs, among existing ones, its popularity served as a sufficient qualification for reference.

The major factors contributing to the computational efficiency of our method are that (i) it fully exploits the structural advantages of NMNFP—the network sparsity and the special type of inequality constraints including coupling capacity constraints, and (ii) it utilizes commodity decomposition and arc decomposition to resolve the potential numerical difficulties caused by the large dimension.

Our method has potential application to other network problems, and it is very likely that we can enlarge the applications of the successive projection and (truncated) seal algorithm to a larger class of projection problems. Furthermore, the efficiency of our method can be increased further if parallel processors are available.

APPENDIX

We prove Theorem 3 using the induction rule. However, the proof is rather complicated, so we will proceed step by step and concatenate all the results at the end. For the sake of simplicity, we omit $(\lambda(t))$ in $\hat{d}f_a(\lambda(t))$ and $\tilde{d}f(\lambda(t))$.

Lemma 7. *Theorem* 3 *is true for* $K = 2$.

Proof. Consider the constraint set $\mathcal{F}_a - f_a(l)$, as shown in the shaded area of Figure 3. For simplicity, we assume $f_a^k(l) = 0$, for every k. We can make this assumption without loss of generality, because without it we need only change the origin of the coordinate system in Figure 3 and change the intersection points of the slanted line and axes as we have shown in Figure 1. If a point lies in region I, III, or V, the corresponding projection will be i, iii, or v, respectively, as indicated in Figure 3. For points in region II (or IV), the corresponding projection should lie on the edge connecting points i and iii (or the edge connecting points iii and v). For example, ii is the projection of point A onto the edge connecting points i and iii. Similarly, iv is the projection of point B. Thus, by applying our successive projection and (truncated) seal algorithm to a point lying in region I, II, III, IV, or V, we can obtain the corresponding projection at the end of Step SP1 of the first iteration. Furthermore, if a point

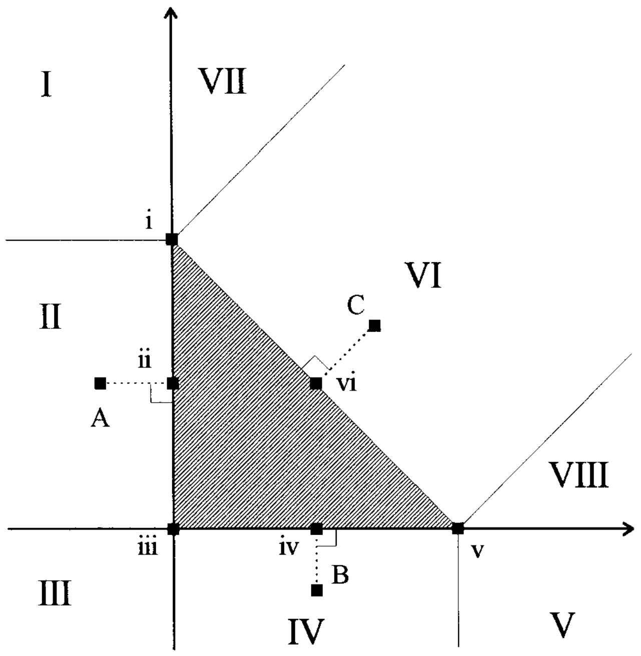

Fig. 3. Illustration of successive projection and (truncated) seal algorithm for the case of $K = 2$.

is in region VI, the corresponding projection should lie on the edge connecting points i and v. For example, point vi is the projection of point C, and we can obtain point vi at the end of Step SP3 in the first iteration and we have $\tau \geq 0$ in Step SP4; therefore, the algorithm stops. The most complicated case occurs when a point lies in region VII (or VIII), for which the corresponding projection is i (or v). In such a case, when we go through Steps SP1 to SP3 in the first iteration, we will obtain a negative value for τ in Step SP4. The algorithm will then carry out a truncation of the values of $\tilde{d}f_a^1$ and $\tilde{d}f_a^2$ obtained from Step SP3 and adjust the set $\mathcal{K}'(a)$ and proceed with the next iteration. After going through Steps SP2 and SP3 in the second iteration, the corresponding projection i (or v) is obtained. This shows that for $K = 2$, at most two iterations of the algorithm are needed to obtain the projection. This completes the proof of this lemma. ■

For the sake of explanation, in the rest of the proof, we assume the following:

Assumption. $f_a^k(l) = 0$, for every k.

As we have stated in the proof of Lemma 7, this assumption does not sacrifice generality. To proceed further, we need to define certain notations: For $K = n$, we let the sets $S_1(n)$, $\cup_{j=1}^n S_2^j(n)$, $S_3(n)$, $\cup_{j=1}^n S_4^j(n)$ be such that $S_1(n) = \{(df_a^1, \ldots, df_a^n) | df_a^k \geq 0, k = 1, \ldots, n$, and $\Sigma_{k=1}^n df_a^k \leq d_a\}$; $S_2^j(n) = \{(df_a^1, \ldots, df_a^n) | df_a^j < 0\}$, $S_3(n) = \{(df_a^1, \ldots, df_a^n) | df_a^k \geq 0$, $k = 1, \ldots, n$, and $\Sigma_{k=1}^n df_a^k > d_a$, $-(n-1)df_a^j + \Sigma_{k\neq j} df_a^k \leq d_a$, $j = 1, \ldots, n\}$, and $S_4^j(n) = \{(df_a^1, \ldots, df_a^n) | df_a^k \geq 0$, $k = 1, \ldots, n$, and $\Sigma_{k=1}^n df_a^k > d_a$, $-(n-1)df_a^j + \Sigma_{k\neq j} df_a^k > d_a\}$. Then, $S_1(n) \cup (\cup_{j=1}^n S_2^j(n)) \cup S_3(n) \cup (\cup_{j=1}^n S_4^j(n)) = \Re^n$. Note that $S_1(n)$ denotes the constraint set $\mathcal{F}_a - f_a(l)$ on our projection problem under the assumption that $f_a^k(l) = 0$ for every k. To examine the above four subsets, we can take the case of $n = 2$ shown in Figure 3 as an example. The shaded area in Figure 3 corresponds to the set $S_1(2)$. The set $\cup_{j=1}^2 S_2^j(2)$ corresponds to the union of the regions I, II, III, IV, and V. The set $S_3(2)$ corresponds to region VI, and the set $\cup_{j=1}^2 S_4^j(2)$ corresponds to the union of regions VII and VIII. Thus, any point $(df_a^1, \ldots, df_a^n) \in \Re^n$ must belong to one of the above four subsets.

Lemma 8. *If Theorem* 3 *is true for* $K = n - 1$ *and if* $(\tilde{d}f_a^1, \ldots, \tilde{d}f_a^n) \in \cup_{j=1}^n S_2^j(n)$, *then Theorem* 3 *is true for* $K = n$.

Proof. If the point $(\tilde{d}f_a^1, \ldots, \tilde{d}f_a^n) \in \cup_{j=1}^n S_2^j(n)$, then there must exist a j, say n, such that $\tilde{d}f_a^n < 0$. Then, at the end of Step SP1 of the first iteration $\tilde{d}f_a^n$ is set at 0, because we have assumed that $f_a^k(l) = 0$ for every k. We claim the following: If $(\hat{d}f_a^1, \ldots, \hat{d}f_a^{n-1})$ is the projection of $(\tilde{d}f_a^1, \ldots, \tilde{d}f_a^{n-1})$ in the case of $K = n - 1$, then $(\hat{d}f_a^1, \ldots, \hat{d}f_a^{n-1}, 0)$ is the projection of $(\tilde{d}f_a^1, \ldots, \tilde{d}f_a^n)$ onto the set $S_1(n)$.

We will prove the above claim by contradiction. Suppose that the claim is false. We assume that $(df_a^{1*}, \ldots, df_a^{n*}) \in S_1(n)$ is the projection and $(df_a^{1*}, \ldots, df_a^{n*}) \neq (\hat{d}f_a^1, \ldots, \hat{d}f_a^{n-1}, 0)$. Then, df_a^{n*} must be 0, because if $df_a^{n*} > 0$, then since $\tilde{d}f_a^n < 0$, we have $\|(\tilde{d}f_a^1, \ldots, \tilde{d}f_a^n) - (df_a^{1*}, \ldots, df_a^{n*})\|_2^2 > \|(\tilde{d}f_a^1, \ldots, \tilde{d}f_a^n) - (df_a^{1*}, \ldots, df_a^{n-1*}, 0)\|_2^2$. This contradicts $(df_a^{1*}, \ldots, df_a^{n-1*}, df_a^{n*})$ being the projection. We may write $(\tilde{d}f_a^1, \ldots, \tilde{d}f_a^n) - (df_a^{1*}, \ldots, df_a^{n-1*}, 0) = (0, \ldots, 0, \tilde{d}f_a^n) + (\tilde{d}f_a^1 - df_a^{1*}, \ldots, \tilde{d}f_a^{n-1} - df_a^{n-1*}, 0)$. By inspection, $(0, \ldots, 0, \tilde{d}f_a^n)$ and $(\tilde{d}f_a^1 - df_a^{1*}, \ldots, \tilde{d}f_a^{n-1} - df_a^{n-1*}, 0)$ are orthogonal. Thus,

$$\begin{aligned}\|(\tilde{d}f_a^1, \ldots, \tilde{d}f_a^{n-1}, \tilde{d}f_a^n) &- (df_a^{1*}, \ldots, df_a^{n-1*}, 0)\|_2^2 \\ &= \|(0, \ldots, 0, \tilde{d}f_a^n)\|_2^2 \\ &+ \|(\tilde{d}f_a^1 - df_a^{1*}, \ldots, \tilde{d}f_a^{n-1} - df_a^{n-1*}, 0)\|_2^2.\end{aligned} \tag{42}$$

Furthermore,

$$\begin{aligned}\|(\tilde{d}f_a^1, \ldots, \tilde{d}f_a^{n-1}, \tilde{d}f_a^n) &- (\hat{d}f_a^1, \ldots, \hat{d}f_a^{n-1}, 0)\|_2^2 \\ &= \|(0, \ldots, 0, \tilde{d}f_a^n)\|_2^2 \\ &+ \|(\tilde{d}f_a^1 - \hat{d}f_a^1, \ldots, \tilde{d}f_a^{n-1} - \hat{d}f_a^{n-1}, 0)\|_2^2,\end{aligned} \tag{43}$$

because $(0, \ldots, 0, \tilde{d}f_a^n)$ and $(\tilde{d}f_a^1 - \hat{d}f_a^1, \ldots, \tilde{d}f_a^{n-1} - \hat{d}f_a^{n-1}, 0)$ are also orthogonal. By assumption, $(df_a^{1*}, \ldots, df_a^{n-1*}, 0)$ is the projection of $(\tilde{d}f_a^1, \ldots, \tilde{d}f_a^n)$; then,

$$\begin{aligned}\|(\tilde{d}f_a^1, \ldots, \tilde{d}f_a^n) &- (df_a^{1*}, \ldots, df_a^{n-1*}, 0)\|_2^2 \\ < \|(\tilde{d}f_a^1, \ldots, \tilde{d}f_a^n) &- (\hat{d}f_a^1, \ldots, \hat{d}f_a^{n-1}, 0)\|_2^2.\end{aligned} \tag{44}$$

Then, from (42) and (43), the inequality (44) becomes

$$\begin{aligned}\|(\tilde{d}f_a^1 - df_a^{1*}, \tilde{d}f_a^2 - df_a^{2*}, \ldots, &\tilde{d}f_a^{n-1} - df_a^{n-1*}, 0)\|_2^2 \\ < \|(\tilde{d}f_a^1 - \hat{d}f_a^1, \tilde{d}f_a^2 - \hat{d}f_a^2, \ldots, &\tilde{d}f_a^{n-1} - \hat{d}f_a^{n-1}, 0)\|_2^2,\end{aligned} \tag{45}$$

implying that

$$\|(\tilde{d}f_a^1 - df_a^{1*}, \ldots, \tilde{d}f_a^{n-1} - df_a^{n-1*})\|_2^2 < \|(\tilde{d}f_a^1 - \hat{d}f_a^1, \ldots, \tilde{d}f_a^{n-1} - \hat{d}f_a^{n-1})\|_2^2. \tag{46}$$

This contradicts the assumption of the claim. Thus, we have proven that $(\hat{d}f_a^1, \ldots, \hat{d}f_a^{n-1}, 0)$ is the projection of $(\tilde{d}f_a^1, \ldots, \tilde{d}f_a^n)$ onto $S_1(n)$.

By assumption of this lemma, it takes at most $n-1$ iterations to obtain the projection $(\hat{d}f_a^1, \ldots, \hat{d}f_a^{n-1})$ starting from the point $(\tilde{d}f_a^1, \ldots, \tilde{d}f_a^{n-1})$. Thus, it takes at most n iterations to obtain the projection $(\hat{d}f_a^1, \ldots, \hat{d}f_a^{n-1}, 0)$. This completes the proof. ■

Lemma 9. *If Theorem 3 is true for $K = n - 1$ and if the point $(\tilde{d}f_a^1, \ldots, \tilde{d}f_a^n) \in S_3(n)$ then Theorem 3 is true for $K = n$.*

Proof. If the point $(\tilde{d}f_a^1, \ldots, \tilde{d}f_a^n) \in S_3(n)$, then, in the first iteration, the algorithm skips Step SP1 and goes through Steps SP2 and SP3. The resulting point becomes $(\tilde{d}f_a^1 + \eta, \ldots, \tilde{d}f_a^n + \eta)$, where $\eta = (d_a - \Sigma_{k=1}^n \tilde{d}f_a^k)/n$. By the definition of $S_3(n)$, we have

$$-(n-1)\tilde{d}f_a^j + \sum_{k\neq j} \tilde{d}f_a^k \leq d_a, \quad j = 1, \ldots, n,$$

which implies that

$$\tilde{d}f_a^j + \frac{d_a - \sum\limits_{k=1}^{n} \tilde{d}f_a^k}{n} \geq 0, \quad j = 1, \ldots, n; \tag{47}$$

i.e., $\tilde{d}f_a^j + \eta \geq 0$, for every $j = 1, 2, \ldots, n$. Furthermore, from the value of η, we can obtain

$$\sum_{k=1}^{n} (\tilde{d}f_a^k + \eta) = d_a. \tag{48}$$

Combining (47) and (48) shows that the point $(\tilde{d}f_a^1 + \eta, \ldots, \tilde{d}f_a^n + \eta)$ is in $S_1(n)$ and on the surface $\Sigma_{k=1}^n df_a^k = d_a$. Furthermore, the vector $(\eta, \eta, \ldots, \eta)$, which is $(\tilde{d}f_a^1 + \eta, \ldots, \tilde{d}f_a^n + \eta) - (\tilde{d}f_a^1, \ldots, \tilde{d}f_a^n)$, is perpendicular to the surface $\Sigma_{k=1}^n df_a^k = d_a$. Thus, the point $(\tilde{d}f_a^1 + \eta, \ldots, \tilde{d}f_a^n + \eta)$ has the shortest distance from $(\tilde{d}f_a^1, \ldots, \tilde{d}f_a^n)$ among all points in $S_1(n)$. Thus, it takes only one iteration of the algorithm to complete this projection. This completes the proof. ■

The case for the point $(\tilde{d}f_a^1, \ldots, \tilde{d}f_a^n) \in \cup_{j=1}^n S_4^j(n)$ is more complicated than are the cases in the preceding two lemma proofs. As mentioned previously, the region represented by $\cup_{j=1}^n S_4^j(n)$ when $n = 2$ is the union of regions VII and VIII in Figure 3. In the proof of Lemma 7, we described how the truncated seal step is applied to points lying in these regions. The following five lemmas (Lemmas 10–14) will generalize this case from $n = 2$ in Lemma 7 to a general n.

Lemma 10. *If $(\tilde{d}f_a^1, \ldots, \tilde{d}f_a^n) \in \cup_{j=1}^n S_4^j(n)$, then at the end of the first iteration, the resulting point will be $(\hat{d}f_a^1 - \hat{d}f_a^l, \ldots, \hat{d}f_a^{l-1} - \hat{d}f_a^l, 0, \hat{d}f_a^{l+1} - \hat{d}f_a^l, \ldots, \hat{d}f_a^n - \hat{d}f_a^l)$, where*

$$\hat{d}f^k = \tilde{d}f_a^k + \frac{d_a - \sum\limits_{j=1}^{n} \tilde{d}f_a^j}{n}, \tag{49}$$

and $\hat{d}f^l$ is the most negative component among all $\hat{d}f^k$, $k = 1, 2, \ldots, n$. Without loss of generality, we may assume that $l = n$; then, at the end of the first iteration, the resulting point is $(\hat{d}f_a^1 - \hat{d}f_a^n, \ldots, \hat{d}f_a^{n-1} - \hat{d}f_a^n, 0)$. Furthermore, the points $(\tilde{d}f_a^1, \ldots, \tilde{d}f_a^n)$, $(\hat{d}f_a^1, \ldots, \hat{d}f_a^n)$, and $(\hat{d}f_a^1 - \hat{d}f_a^n, \ldots, \hat{d}f_a^{n-1} - \hat{d}f_a^n, 0)$ are on the surface $-(n-1)df_a^n + \Sigma_{k\neq n} df_a^k = C_a$ for some $C_a > d_a$.

Proof. If $(\tilde{d}f_a^1, \ldots, \tilde{d}f_a^n) \in \cup_{j=1}^n S_4^j(n)$, then $(\tilde{d}f_a^1, \ldots, \tilde{d}f_a^n) \in S_4^j(n)$ for some j. Therefore,

$$-(n-1)\tilde{d}f_a^j + \sum_{k\neq j} \tilde{d}f_a^k > d_a. \tag{50}$$

Then, after going through Step SP3 of the first iteration, we have that

$$\hat{d}f_a^k = \tilde{d}f_a^k + \frac{d_a - \sum\limits_{j=1}^{n} \tilde{d}f_a^j}{n}, \quad k = 1, \ldots, n, \tag{51}$$

which is (49). Then, from (50), we see that $\hat{d}f_a^j < 0$. Assuming that $\hat{d}f_a^l$ is the most negative component among $\hat{d}f_a^k$, $k = 1, \ldots, n$, then at the end of Step SP4 of the first iteration, the resulting point is $(\hat{d}f_a^1 - \hat{d}f_a^l, \ldots, \hat{d}f_a^{l-1} - \hat{d}f_a^l, 0, \hat{d}f_a^{l+1} - \hat{d}f_a^l, \ldots, \hat{d}f_a^n - \hat{d}f_a^l)$. Assuming that $l = n$, we have the resulting point as $(\hat{d}f_a^1 - \hat{d}f_a^n, \ldots, \hat{d}f_a^{n-1} - \hat{d}f_a^n, 0)$, and $\hat{d}f_a^n < 0$; then, from (51),

$$\hat{d}f_a^n = \tilde{d}f_a^n + \frac{d_a - \sum\limits_{k=1}^{n} \tilde{d}f_a^k}{n} < 0, \tag{52}$$

which implies that

$$-(n-1)\tilde{d}f_a^n + \sum_{k\neq n} \tilde{d}f_a^k > d_a. \tag{53}$$

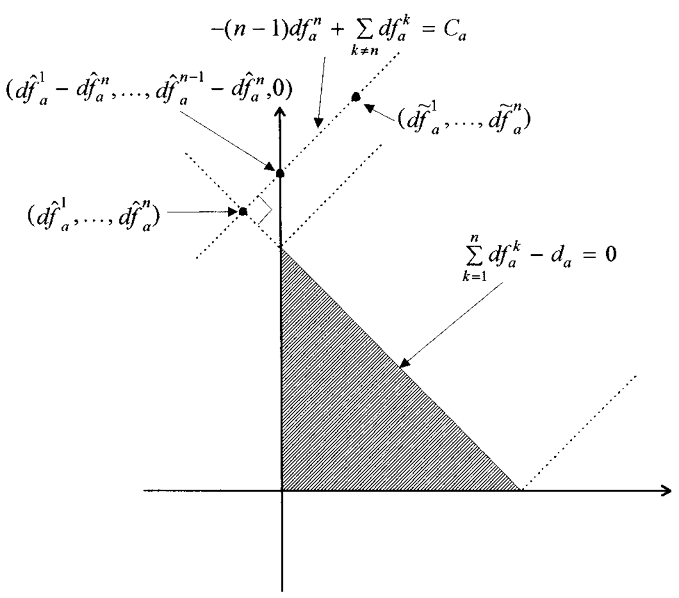
$-(n-1)df_a^n + \sum_{k\neq n} df_a^k = C_a$
$(\hat{df}_a^1 - \hat{df}_a^n, \ldots, \hat{df}_a^{n-1} - \hat{df}_a^n, 0)$
$(\tilde{df}_a^1, \ldots, \tilde{df}_a^n)$
$(\hat{df}_a^1, \ldots, \hat{df}_a^n)$
$\sum_{k=1}^n df_a^k - d_a = 0$

Fig. 4. Illustration of geometrical relationships of points $(\tilde{d}f_a^1, \ldots, \tilde{d}f_a^n)$, $(\hat{d}f_a^1, \ldots, \hat{d}f_a^n)$, and $(\hat{d}f_a^1 - \hat{d}f_a^n, \ldots, \hat{d}f_a^{n-1} - \hat{d}f_a^n, 0)$, and the surfaces $-(n-1)df_a^n + \Sigma_{k\neq n} df_a^k = C_a$ and $\Sigma_{k=1}^n df_a^k - d_a = 0$.

Therefore,

$$-(n-1)\hat{d}f_a^n + \sum_{k\neq n} \hat{d}f_a^k = C_a \quad \text{for some} \quad C_a > d_a. \tag{54}$$

From (51) and (54), we have

$$-(n-1)\tilde{d}f_a^n + \sum_{k\neq n} \tilde{d}f_a^k = C_a. \tag{55}$$

Thus, both $(\tilde{d}f_a^1, \ldots, \tilde{d}f_a^n)$ and $(\hat{d}f_a^1, \ldots, \hat{d}f_a^n)$ are on the surface $-(n-1)df_a^n + \Sigma_{k\neq n} df_a^k = C_a$. Furthermore,

$$-(n-1)\cdot 0 + \sum_{k\neq n} (\hat{d}f_a^k - \hat{d}f_a^n) = -(n-1)\hat{d}f_a^n + \sum_{k\neq n} \hat{d}f_a^k$$

shows that $(\hat{d}f_a^1 - \hat{d}f_a^n, \ldots, \hat{d}f_a^{n-1} - \hat{d}f_a^n, 0)$ is also on the surface $-(n-1)df_a^n + \Sigma_{k\neq n} df_a^k = C_a$. This completes the proof. ■

Figure 4 shows the geometrical relationships of the points $(\tilde{d}f_a^1, \ldots, \tilde{d}f_a^n)$, $(\hat{d}f_a^1, \ldots, \hat{d}f_a^1)$, and $(\hat{d}f_a^1 - \hat{d}f_a^n, \ldots, \hat{d}f_a^{n-1} - \hat{d}f_a^n, 0)$, and the surfaces $-(n-1)df_a^n + \Sigma_{k\neq n} df_a^k = C_a$ and $\Sigma_{k=1}^n df_a^k - d_a = 0$. Furthermore, it can be seen from Figure 4 that the surfaces $-(n-1)df_a^n + \Sigma_{k\neq n} df_a^k = C_a$ and $\Sigma_{k=1}^n df_a^k - d_a = 0$ are perpendicular to each other.

Lemma 11. *Suppose that the point* $(\tilde{d}f_a^1, \ldots, \tilde{d}f_a^n) \in \cup_{j=1}^n S_4^j(n)$; *then, if* $(df_a^{1*}, \ldots, df_a^{n-1*})$ *is the projection of* $(\hat{d}f_a^1 - \hat{d}f_a^n, \ldots, \hat{d}f_a^{n-1} - \hat{d}f_a^n)$ *in the case of* $K = n-1$, (*i*) *both* $(df_a^{1*}, \ldots, df_a^{n-1*}, 0)$ *and* $(\hat{d}f_a^1, \ldots, \hat{d}f_a^{n-1}, \hat{d}f_a^n)$ *are on the surface* $\Sigma_{k=1}^n df_a^k - d_a = 0$, *and* (*ii*) $\|(df_a^{1*}, \ldots, df_a^{n-1*}, 0) - (\hat{d}f_a^1 - \hat{d}f_a^n, \ldots, \hat{d}f_a^{n-1} - \hat{d}f_a^n, 0)\|_2^2 = \|(df_a^{1*}, \ldots, df_a^{n-1*}, 0) - (\hat{d}f_a^1, \ldots, \hat{d}f_a^n)\|_2^2 + \|(\hat{d}f_a^n, \ldots, \hat{d}f_a^n)\|_2^2$.

Proof. We see that $(df_a^{1*}, \ldots, df_a^{n-1*}, 0)$ must be on the surface $\Sigma_{k=1}^n df_a^k - d_a = 0$, because, otherwise, $\Sigma_{k=1}^{n-1} df_a^{k*} + 0 < d_a$, the intersection point of the surface $\Sigma_{k=1}^n df_a^k - d_a = 0$ and the line connecting points $(df_a^{1*}, \ldots, df_a^{n-1*}, 0)$ and $(\hat{d}f_a^1 - \hat{d}f_a^n, \ldots, \hat{d}f_a^{n-1} - \hat{d}f_a^n, 0)$ is a shorter distance from $(\hat{d}f_a^1 - \hat{d}f_a^n, \ldots, \hat{d}f_a^{n-1} - \hat{d}f_a^n, 0)$. This contradicts the assumption of this lemma.

From (49) of Lemma 10 and from Figure 4, we see that $(\hat{d}f_a^1, \ldots, \hat{d}f_a^{n-1}, \hat{d}f_a^n)$ is on the surface $\Sigma_{k=1}^K df_a^k - d_a = 0$; it then follows that both $(df_a^{1*}, \ldots, df_a^{n-1*}, 0)$ and $(\hat{d}f_a^1, \ldots, \hat{d}f_a^{n-1}, \hat{d}f_a^n)$ are on the surface $\Sigma_{k=1}^n df_a^k - d_a = 0$. This proves (i).

We can write

$$\begin{aligned}(df_a^{1*}, \ldots, df_a^{n-1*}, 0) &- (\hat{d}f_a^1 - \hat{d}f_a^n, \ldots, \hat{d}f_a^{n-1} - \hat{d}f_a^n, 0)\\ &= [(df_a^{1*} - \hat{d}f_a^1, \ldots, df_a^{n-1*} - \hat{d}f_a^{n-1}, -\hat{d}f_a^n)]\\ &\quad + [(\hat{d}f_a^n, \hat{d}f_a^n, \ldots, \hat{d}f_a^n)].\end{aligned} \tag{56}$$

Because the surfaces $\Sigma_{k=1}^n df_a^k - d_a = 0$ and $-(n-1)df_a^n + \Sigma_{k=1}^{n-1} df_a^k = C_a$ are perpendicular to each other, as shown in Figure 4, and $(\hat{d}f_a^n, \hat{d}f_a^n, \ldots, \hat{d}f_a^n) = (\hat{d}f_a^1, \ldots, \hat{d}f_a^n) - (\hat{d}f_a^1 - \hat{d}f_a^n, \ldots, \hat{d}f_a^{n-1} - \hat{d}f_a^n, 0)$, then from (i) and Lemma 10, $(df_a^{1*} - \hat{d}f_a^1, \ldots, df_a^{n-1*} - \hat{d}f_a^n, -\hat{d}f_a^n)$ is orthogonal to $(\hat{d}f_a^n, \hat{d}f_a^n, \ldots, \hat{d}f_a^n)$.

From (56), we then have

$$\begin{aligned}\|(df_a^{1*}, \ldots, df_a^{n-1*}, 0) &- (\hat{d}f_a^1 - \hat{d}f_a^n, \ldots, \hat{d}f_a^{n-1} - \hat{d}f_a^n, 0)\|_2^2\\ = \|(df_a^{1*}, \ldots, df_a^{n-1*}, 0) &- (\hat{d}f_a^1, \ldots, \hat{d}f_a^n)\|_2^2 + \|\hat{d}f_a^n, \ldots, \hat{d}f_a^n)\|_2^2.\end{aligned} \tag{57}$$

This proves (ii).

Lemma 12. *Suppose that the point* $(\tilde{d}f_a^1, \ldots, \tilde{d}f_a^n) \in \cup_{j=1}^n S_4^j(n)$; then, if $(df_a^{1*}, \ldots, df_a^{n-1*})$ *is the projection of* $(\hat{d}f_a^1 - \hat{d}f_a^n, \ldots, \hat{d}f_a^{n-1} - \hat{d}f_a^n)$ *in the case of* $K = n - 1$, $(df_a^{1*}, \ldots, df_a^{n-1*}, 0)$ *is the projection of* $(\hat{d}f_a^1, \ldots, \hat{d}f_a^{n-1}, \hat{d}f_a^n)$ *in the case of* $K = n$.

Proof. We will prove this lemma by contradiction. Suppose that there exists another point $(\bar{d}f_a^1, \ldots, \bar{d}f_a^{n-1}, \bar{d}f_a^n) \in S_1(n)$ and $(\bar{d}f_a^1, \ldots, \bar{d}f_a^{n-1}, \bar{d}f_a^n) \neq (df_a^{1*}, \ldots, df_a^{n-1*}, 0)$ such that

$$\begin{aligned}
&\|(\bar{d}f_a^1, \ldots, \bar{d}f_a^n) - (\hat{d}f_a^1, \ldots, \hat{d}f_a^n)\|_2 \\
&\quad < \|(df_a^1, \ldots, df_a^n) - (\hat{d}f_a^1, \ldots, \hat{d}f_a^n)\|_2 \\
&\qquad \text{for every} \quad (df_a^1, \ldots, df_a^n) \in S_1(n) \\
&\qquad \text{and} \quad (df_a^1, \ldots, df_a^n) \neq (\bar{d}f_a^1, \ldots, \bar{d}f_a^n).
\end{aligned} \tag{58}$$

We see that $\bar{d}f_a^n$ must equal 0, because if $\bar{d}f_a^n > 0$, then, since $\hat{d}f_a^n < 0$, the intersection point of the set $S_1(n)$ and the line connecting the points $(\hat{d}f_a^1, \ldots, \hat{d}f_a^{n-1}, \hat{d}f_a^n)$ and $(\bar{d}f_a^1, \ldots, \bar{d}f_a^{n-1}, \bar{d}f_a^n)$ is a shorter distance from $(\hat{d}f_a^1, \ldots, \hat{d}f_a^{n-1}, \hat{d}f_a^n)$. This contradicts (58); therefore, $\bar{d}f_a^n = 0$. From (i) of Lemma 11, $(\hat{d}f_a^1, \ldots, \hat{d}f_a^n)$ is on the surface $\Sigma_{k=1}^n df_a^k - d_a = 0$, $(\bar{d}f_a^1, \ldots, \bar{d}f_a^{n-1}, 0)$ must also be on the surface $\Sigma_{k=1}^n df_a^k - d_a = 0$; otherwise, it contradicts (58). Since both $(\bar{d}f_a^1, \ldots, \bar{d}f_a^{n-1}, 0)$ and $(\hat{d}f_a^1, \ldots, \hat{d}f_a^n)$ are on the surface $\Sigma_{k=1}^n df_a^k - d_a = 0$, and by Lemma 10 that both $(\hat{d}f_a^1, \ldots, \hat{d}f_a^{n-1}, \hat{d}f_a^n)$ and $(\hat{d}f_a^1 - \hat{d}f_a^n, \ldots, \hat{d}f_a^{n-1} - \hat{d}f_a^n, 0)$ are on the surface $-(n-1)df_a^n + \Sigma_{k\neq n} df_a^k = C_a$, then $(\bar{d}f_a^1, \ldots, \bar{d}f_a^{n-1}, 0) - (\hat{d}f_a^1, \ldots, \hat{d}f_a^{n-1}, \hat{d}f_a^n)$ is orthogonal to $(\hat{d}f_a^n, \hat{d}f_a^n, \ldots, \hat{d}f_a^n)$ because the surfaces $\Sigma_{k=1}^n df_a^k - d_a = 0$ and $-(n-1)df_a^n + \Sigma_{k=1}^{n-1} df_a^k = C_a$ are perpendicular to each other. Thus, we have

$$\begin{aligned}
&\|(\bar{d}f_a^1, \ldots, \bar{d}f_a^{n-1}, 0) \\
&\quad - (\hat{d}f_a^1 - \hat{d}f_a^n, \ldots, \hat{d}f_a^1 - \hat{d}f_a^{n-1}, 0)\|_2^2 \\
&\quad = \|(\bar{d}f_a^1, \ldots, \bar{d}f_a^{n-1}, 0) \\
&\quad - (\hat{d}f_a^1, \ldots, \hat{d}f_a^n)\|_2^2 \\
&\qquad\qquad + \|(\hat{d}f_a^n, \ldots, \hat{d}f_a^n)\|_2^2.
\end{aligned} \tag{59}$$

From (58), we have

$$\begin{aligned}
&\|(\bar{d}f_a^1, \ldots, \bar{d}f_a^{n-1}, 0) \\
&\quad - (\hat{d}f_a^1, \ldots, \hat{d}f_a^{n-1}, \hat{d}f_a^n)\|_2^2 \\
&\quad < \|(\hat{d}f_a^{1*}, \ldots, \hat{d}f_a^{n-1*}, 0) \\
&\qquad\qquad - (\hat{d}f_a^1, \ldots, \hat{d}f_a^{n-1}, \hat{d}f_a^n)\|_2.
\end{aligned} \tag{60}$$

Adding $\|(\hat{d}f_a^1, \ldots, \hat{d}f_a^n)\|_2^2$ to both sides of (60), we obtain

$$\begin{aligned}
&\|(\bar{d}f_a^1, \ldots, \bar{d}f_a^{n-1}, 0) - (\hat{d}f_a^1, \ldots, \hat{d}f_a^n)\|_2^2 \\
&\quad + \|(\hat{d}f_a^n, \ldots, \hat{d}f_a^n)\|_2^2 \\
&\quad < \|(df_a^{1*}, \ldots, df_a^{n-1*}, 0) \\
&\quad - (\hat{d}f_a^1, \ldots, \hat{d}f_a^{n-1}, \hat{d}f_a^n, 0)\|_2^2 \\
&\qquad\qquad + \|(\hat{d}f_a^n, \ldots, \hat{d}f_a^n)\|_2^2.
\end{aligned} \tag{61}$$

From (ii) of Lemma 11 and (59), we obtain the following from (61)

$$\begin{aligned}
&\|(\bar{d}f_a^1, \ldots, \bar{d}f_a^{n-1}, 0) \\
&\quad - (\hat{d}f_a^1 - \hat{d}f_a^n, \ldots, \hat{d}f_a^{n-1} - \hat{d}f_a^n, 0)\|_2^2 \\
&\quad < \|(df_a^{1*}, \ldots, df_a^{n-1*}, 0) \\
&\quad - (\hat{d}f_a^1 - \hat{d}f_a^n, \ldots, \hat{d}f_a^{n-1} - \hat{d}f_a^n, 0)\|_2^2,
\end{aligned} \tag{62}$$

which implies that

$$\begin{aligned}
&\|(\bar{d}f_a^1, \ldots, \bar{d}f_a^{n-1}) \\
&\quad - (\hat{d}f_a^1 - \hat{d}f_a^n, \ldots, \hat{d}f_a^{n-1} - \hat{d}f_a^n)\|_2^2 \\
&\quad < \|(df_a^{1*}, \ldots, df_a^{n-1*}) \\
&\qquad - (\hat{d}f_a^1 - \hat{d}f_a^n, \ldots, \hat{d}f_a^{n-1} - \hat{d}f_a^n)\|_2.
\end{aligned} \tag{63}$$

Inequality (63) contradicts the assumption of this lemma. This completes the proof. ■

Lemma 13. *Suppose that the point* $(\tilde{d}f_a^1, \ldots, \tilde{d}f_a^n) \in \cup_{j=1}^n S_4^j(n)$, then if $(df_a^{1*}, \ldots, df_a^{n-1*})$ *is the projection of* $(\hat{d}f_a^1 - \hat{d}f_a^n, \ldots, \hat{d}f_a^{n-1} - \hat{d}f_a^n)$ *in the case of* $K = n - 1$, $(df_a^{1*}, \ldots, df_a^{n-1*}, 0)$ *is the projection of* $(\tilde{d}f_a^1, \ldots, \tilde{d}f_a^{n-1}, \tilde{d}f_a^n)$.

Proof. We prove this lemma by contradiction. Suppose that $(\check{d}f_a^1, \ldots, \check{d}f_a^n)$ is the projection of $(\tilde{d}f_a^1, \ldots, \tilde{d}f_a^n)$ and $(\check{d}f_a^1, \ldots, \check{d}f_a^n) \neq (df_a^{1*}, \ldots, df_a^{n-1*}, 0)$. By the definition of $S_4^j(n)$, $\Sigma_{k=1}^n \tilde{d}f_a^k > d_a$, thus, $(\check{d}f_a^1, \ldots, \check{d}f_a^n)$ must be on the surface $\Sigma_{k=1}^n df_a^k - d_a = 0$ because, otherwise, the intersection point of the surface $\Sigma_{k=1}^n df_a^k - d_a = 0$ and the line segment connecting points $(\tilde{d}f_a^1, \ldots, \tilde{d}f_a^n)$ and $(\check{d}f_a^1, \ldots, \check{d}f_a^n)$ is a shorter distance from $(\tilde{d}f_a^1, \ldots, \tilde{d}f_a^n)$. Then, from (i) Lemma 11 and previous statements, both $(\hat{d}f_a^1, \ldots, \hat{d}f_a^n)$ and $(\check{d}f_a^1, \ldots, \check{d}f_a^n)$ are on the surface $\Sigma_{k=1}^n df_a^k - d_a = 0$. From Lemma 10, both $(\tilde{d}f_a^1, \ldots, \tilde{d}f_a^n)$ and $(\hat{d}f_a^1, \ldots, \hat{d}f_a^n)$ are on the surface $-(n-1)df_a^n + \Sigma_{k\neq n} df_a^k = C_a$. Therefore, $(\check{d}f_a^1 - \hat{d}f_a^1, \ldots, \check{d}f_a^n - \hat{d}f_a^n)$ is orthogonal to $(\hat{d}f_a^1 - \tilde{d}f_a^1, \ldots, \hat{d}f_a^n - \tilde{d}f_a^n)$. We can write $(\check{d}f_a^1, \ldots, \check{d}f_a^n) - (\tilde{d}f_a^1,$

$\ldots, \tilde{d}f^{n}_{a}) = [(\check{d}f^{1}_{a}, \ldots, \check{d}f^{n}_{a}) - (\hat{d}f^{1}_{a}, \ldots, \hat{d}f^{n}_{a})] + [(\hat{d}f^{1}_{a}, \ldots, \hat{d}f^{n}_{a}) - (\tilde{d}f^{1}_{a}, \ldots, \tilde{d}f^{n}_{a})]$. Consequently,

$$\begin{aligned} &\|(\check{d}f^{1}_{a}, \ldots, \check{d}f^{n}_{a}) - (\tilde{d}f^{1}_{a}, \ldots, \tilde{d}f^{n}_{a})\|_2^2 \\ &\quad = \|(\check{d}f^{1}_{a}, \ldots, \check{d}f^{n}_{a}) - (\hat{d}f^{1}_{a}, \ldots, \hat{d}f^{n}_{a})\|_2^2 \\ &\qquad + \|(\hat{d}f^{1}_{a}, \ldots, \hat{d}f^{n}_{a}) - (\tilde{d}f^{1}_{a}, \ldots, \tilde{d}f^{n}_{a})\|_2^2. \end{aligned} \tag{64}$$

From (i) of Lemma 11, $(df^{1*}_{a}, \ldots, df^{n-1*}_{a}, 0)$ is on the surface $\Sigma_{k=1}^{n} df^{k}_{a} - d_a = 0$, so that $(df^{1*}_{a}, \ldots, df^{n-1*}_{a}, 0) - (\hat{d}f^{1}_{a}, \ldots, \hat{d}f^{n-1}_{a}, \hat{d}f^{n}_{a})$ is also orthogonal to $(\hat{d}f^{1}_{a}, \ldots, \hat{d}f^{n}_{a}) - (\tilde{d}f^{1}_{a}, \ldots, \tilde{d}f^{n}_{a})$. Thus,

$$\begin{aligned} &\|(df^{1*}_{a}, \ldots, df^{n-1*}_{a}, 0) - (\tilde{d}f^{1}_{a}, \ldots, \tilde{d}f^{n}_{a})\|_2^2 \\ &\quad = \|(df^{1*}_{a}, \ldots, df^{n-1*}_{a}, 0) \\ &\qquad - (\hat{d}f^{1}_{a}, \ldots, \hat{d}f^{n}_{a})\|_2^2 \\ &\qquad + \|(\hat{d}f^{1}_{a}, \ldots, \hat{d}f^{n}_{a}) - (\tilde{d}f^{1}_{a}, \ldots, \tilde{d}f^{n}_{a})\|_2^2. \end{aligned} \tag{65}$$

From Lemma 12, we have

$$\begin{aligned} &\|(df^{1*}_{a}, \ldots, df^{n-1*}_{a}, 0) - (\hat{d}f^{1}_{a}, \ldots, \hat{d}f^{n}_{a})\|_2^2 \\ &\qquad < \|(\check{d}f^{1}_{a}, \ldots, \check{d}f^{n-1}_{a}, \check{d}f^{n}_{a}) \\ &\qquad\qquad - (\hat{d}f^{1}_{a}, \ldots, \hat{d}f^{n}_{a})\|_2^2. \end{aligned} \tag{66}$$

Then, from (64) and (65), the following inequality can be obtained from (66):

$$\begin{aligned} &\|(df^{1*}_{a}, \ldots, df^{n-1*}_{a}, 0) - (\tilde{d}f^{1}_{a}, \ldots, \tilde{d}f^{n}_{a})\|_2^2 \\ &\qquad < \|(\check{d}f^{1}_{a}, \ldots, \check{d}fa^{n}) - (\tilde{d}f^{1}_{a}, \ldots, \tilde{d}f^{n}_{a})\|_2^2, \end{aligned}$$

which contradicts the assumption for $(\check{d}f^{1}_{a}, \ldots, \check{d}f^{n}_{a})$. This completes the proof. ■

Lemma 14. *If Theorem* 3 *is true for* $K = n - 1$ *and if* $(\tilde{d}f^{1}_{a}, \ldots, \tilde{d}f^{n}_{a}) \in \cup_{j=1}^{n} S_4^j(n)$, *then Theorem* 3 *is true for* $K = n$.

Proof. From Lemma 10, at the end of the first iteration, the algorithm obtains the point $(\hat{d}f^{1}_{a} - \hat{d}f^{n}_{a}, \ldots, \hat{d}f^{n-1}_{a} - \hat{d}f^{n}_{a}, 0)$, and the last component is set at 0. We have shown in Lemma 13 that $(df^{1*}_{a}, \ldots, df^{n-1*}_{a}, 0)$ is the projection of $(\tilde{d}f^{1}_{a}, \ldots, \tilde{d}f^{n}_{a})$ provided that $(df^{1*}_{a}, \ldots, df^{n-1*}_{a})$ is the projection of $(\hat{d}f^{1}_{a} - \hat{d}f^{n}_{a}, \ldots, \hat{d}f^{n-1}_{a} - \hat{d}f^{n}_{a})$. Then, from the assumption of this lemma, we can obtain $(df^{1*}_{a}, \ldots, df^{n-1*}_{a})$ in at most $n - 1$ iterations. This shows that we can obtain the projection $(df^{1*}_{a}, \ldots, df^{n-1*}_{a}, 0)$ in at most n iterations. This completes the proof. ■

Now, we are ready to prove Theorem 3.

Proof of Theorem 3. We will prove this theorem by the induction rule:

(i) From Lemma 7, we have shown that this theorem is true for $K = 2$.
(ii) We assume that this theorem is true for the case $K = n - 1$. We shall now show that
(iii) this theorem is true for $K = n$.

Since $S_1(n) \cup (\cup_{j=1}^{n} S_2^j(n)) \cup S_3(n) \cup (\cup_{j=1}^{n} S_4^j(n)) = \Re^n$, we can consider the projection of any point $(\tilde{d}f^{1}_{a}, \ldots, \tilde{d}f^{n}_{a})$ in $\Re^n$ in the following four cases:

CASE (A). For any point that belongs to $S_1(n)$, no projection is needed.

CASE (B). If $(\tilde{d}f^{1}_{a}, \ldots, \tilde{d}f^{n}_{a}) \in \cup_{j=1}^{n} S_2^j(n)$, then from Lemma 8, this theorem is true for $K = n$.

CASE (C). If $(\tilde{d}f^{1}_{a}, \ldots, \tilde{d}f^{n}_{a}) \in S_3(n)$, then from Lemma 9, this theorem is true for $K = n$.

CASE (D). If the point $(\tilde{d}f^{1}_{a}, \ldots, \tilde{d}f^{n}_{a}) \in \cup_{j=1}^{n} S_4^j(n)$, then from Lemma 14, this theorem is true for $K = n$.

Thus, we have completed the proof. ■

The authors wish to thank the anonymous reviewer who has given us a great many valuable, constructive, and instructive comments which helped us to rewrite the paper completely. Especially, the presentation of the proof of Theorem 3 is indebted to him (or her).

REFERENCES

[1] A. A. Assad, Multicommodity network flows—A survey. *Networks* **8** (1978) 37–91.
[2] M. S. Bazaraa and C. M. Shetty, *Nonlinear Programing.* Wiley, New York (1979).
[3] D. P. Bertsekas and E. M. Gafni, Projected newton method and optimization of multicommodity flows. *IEEE Trans. Autom. Contr.* **AC-28** (1983) 1090–1096.
[4] D. P. Bertsekas, P. A. Hossein, and P. Tseng, Relaxation methods for network flow problems with covex arc costs. *SIAM J. Cont. Optim.* **25** (1987) 1219–1243.
[5] D. P. Bertsekas and J. N. Tsitsiklis, *Parallel and Distributed Computation: Numerical Methods.* Prentice-Hall, London (1989).
[6] D. G. Cantor and M. Gerla, Optimal routing in a packet-switched computer network. *IEEE Trans. Comput.* **C-23**(10) (1974) 1062–1068.
[7] W. Chou and H. Frank, Routing strategies for computer network design. *Proceedings of the Computer Communications Networks and Teletraffic* (1972) 301–309.
[8] S. C. Dafermos, An extended traffic assignment model

with applications to two-way traffic. *Trans. Sci.* **5** (1971) 336–389.

[9] S. C. Dafermos and F. T. Sparrow, The traffic assignment problem for a general network. *J. Res. Nat. Bur. Stand. B* (1969) 395–412.

[10] R. S. Dembo and U. Tulowitzki, Computing equilibria on large multicommodity networks: An application of truncated quadratic programming algorithms. *Networks* **18** (1988) 273–284.

[11] H. Frank and W. Chou, Routing in computer networks. *Networks* **1**(2) (1971) 99–112.

[12] L. Fratta, M. Gerla, and L. Kleinrock, The flow deviation method: An approach to store-and-forward communication network design. *Networks* **3**(2) (1973) 97–113.

[13] A. M. Geoffrion and G. W. Graves, Multicommodity distribution system design by benders decomposition. *Mgmt. Sci.* **20**(5) (1974) 822–884.

[14] B. Golden, A minimum-cost multicommodity network flow problem concerning imports and exports. *Networks* **5**(4) (1975) 331–356.

[15] M. A. Hall and E. L. Peterson, Highway traffic equilibria analyzed via geometric programming. *Traffice Equilibrium Methods* (M. A. Florian, Ed.). Springer-Verlag, New York (1976) 53–105.

[16] S. P. Han, A successive projection method. *Math. Prog.* **40** (1988) 1–14.

[17] J. L. Kennington and R. V. Helgason, *Algorithms for network programming.* Wiley, New York (1980).

[18] R. W. Klessing, An algorithm for nonlinear multicommodity flow problems. *Networks* **4**(4) (1974) 343–355.

[19] L. J. Leblanc, E. K. Morlok, and W. P. Pierskalla, An accurate and efficient approach to equilibrium traffic assignment on congested networks. Interactive graphics & trans. systems planning. *Trans. Res. Rec.* **491** (1974) 12–33.

[20] T. Leventhal, G. Nemhauser, and L. Trotter, Jr., A column generation algorithm for optional traffic assignment. *Trans. Sci.* **7**(2) (1973) 168–176.

[21] D. Luenberger, *Linear and nonlinear programming,* 2nd ed. Addison-Wesley Reading, MA (1984).

[22] H. Nagamochi, M. Fukushima, and T. Ibaraki, Relaxation methods for the strictly convex multicommodity flow problem with capacity constraints on individual commodities. *Networks* **20** (1990) 409–426.

[23] S. Nguyen, An algorithm for the traffic assignment problem. *Trans. Sci.* **8**(3) (1974) 203–216.

[24] S. Nguyen, A unified approach to equilibrium methods for traffic assignments. *Traffic Equilibrium Methods* (M. Florian, Ed.). Springer-Verlag, New York (1976) 148–182.

[25] A. Ohuchi and I. Kaji, Lagrangian dual coordinatewise maximization algorithm for network transportation problem with quadratic costs. *Networks* **14** (1984) 515–530.

[26] R. B. Potts and R. M. Oliver, *Flows in transportation networks.* Academic Press, New York, London (1972).

[27] M. Schwartz and C. K. Cheung, The gradient projection algorithm for multiple routing in message-switched networks. *Preprint* (June 1975).

[28] B. Shetty and R. Muthukrishnan, A parallel projection for the multicommodity network model. *J. Opl. Res. Soc.* **41** (1990) 837–842.

[29] J. G. Wardrop, Some theoretical aspects of road traffic research. *Proc. Inst. Civ. Eng.* **2**(1) (1952) 325–378.

[30] B. Yaged, Minimum cost routing for static network models. *Networks* **1**(2) (1971) 139–172.