Distributed Iterative Aggregation Algorithms for Box-Constrained Minimization Problems and Optimal Routing in Data Networks

WEI K. TSAI, MEMBER, IEEE, GARNG HUANG, SENIOR MEMBER, IEEE, JOHN K. ANTONIO, MEMBER, IEEE, AND WEI T. TSAI, MEMBER, IEEE

Abstract—**A new gradient projection algorithm using iterative aggregation and disaggregation is proposed and analyzed for box-constrained minimization problems. In a distributed computation model, the algorithm is shown to converge. As an important application, we also show how the algorithm is applied to optimal routing in a large interconnected data communication network. The aggregation/disaggregation method proposed results in a multilevel hierarchical clustering of a large network, which naturally fits the hierarchical topological structure of large networks. An implementation of the algorithm for a 52-node network shows that the serial version of the algorithm has a savings of 35 percent of the computational time as compared to a path-formulated gradient projection code developed by Bertsekas, Gendron, and Tsai, which is among the fastest existing programs for path-formulated optimal routing.**

I. Introduction

ONE interesting class of optimization problems is the one with nonnegativity constraints (i.e., box-constraints) on the variables. For this class of problems, an efficient descent algorithm is the gradient projection (GP) algorithm of the Goldstein–Levitin–Polyak type [14]. On the other hand, one attractive general methodology of solving large-scale computational problems uses iterative aggregations: the so-called iterative aggregation–disaggregation (IAD) methods by the Soviet researchers [1], [5], [6], [9], also known as the aggregation methods by Miranker and colleagues [3], [12].

IAD methods have been shown to accelerate classical algorithms; for example, the multigrid method [11] for discretized elliptic partial differential equations (PDE's), which is based on a similar idea, is known to be the fastest[1] IAD method and have been applied to linear systems of equations [2], [4], [5], eigenvalue problems [3], and linear, nonlinear, and dynamic programming problems [1], [6]–[8]. Among all these applications, the multigrid method is the most mature and fully developed, while the IAD methods for other application areas need to be more fully developed to realize their potential.

Here we propose an algorithm which is called IAD–GP because it combines both the features of IAD and the GP methods. IAD methods can be further classified into multilevel and one-level methods. Also, the aggregation methods reduce the problem size, and thus reduce the computation time. In this paper, we shall concentrate on the two-level methods [1] since the two-level methods reflect the essence of the multilevel and one-level methods.

As an important application, we also show how the algorithm is applied to optimal routing in large interconnected data communication networks. Since most routing algorithms for data networks (as well as many other GP algorithms) need to operate in a distributed computation environment, we show the convergence of the proposed algorithm in a distributed asynchronous computation model.

The main idea of iterative aggregation methods is to replace the original large-scale problem by a series of aggregate problems whose disaggregated solutions converge to the exact solution. Intuition suggests that by using a macro model, the algorithm is capable of moving a large step, thus accelerating the convergence. In this paper we shall not concern ourselves with the analysis of the convergence rate; related theoretical foundation of the acceleration effects can be found in [2], [3], and [12]. We provide, however, a numerical example in data network routing to demonstrate the acceleration effect.

This paper is organized as follows. In Section II we introduce the box-constrained problem and the distributed computation model. In Section III we discuss the aggregation method and a serial version of the IAD–GP algorithm. The main convergence result of the distributed IAD–GP algorithm is presented in Section IV. In Section V, the optimal routing problem in data networks is formulated and transformed into a box-constrained problem, and the convergence of the IAD–GP algorithm applied to this problem is presented. In Section VI, a class of aggregation methods for the optimal routing problem is presented. Some implementation results are presented in Section VII, followed by conclusions in Section VIII.

II. Box-Constrained Problem and Distributed Computation Model

A. Box-Constrained Optimization Problem

The box-constrained optimization problem (BP) introduced below is a quite useful formulation. Not only is the (BP) itself a meaningful problem, but it is also a transformed multicommodity flow problem which is popular in transportation systems, operations research, and data communication networks. In addition, it is the dual problem of an inequality constrained primal problem. The problem is stated as follows:

$$\min J(x)$$

$$\text{s.t. } x \geq 0$$

where $J:\mathcal{R}^N \rightarrow \mathcal{R}$, and J is assumed to be nonnegative and

Manuscript received November 25, 1987; revised July 17, 1988. Paper recommended by Associate Editor, S. Verdu. This work was supported in part by Texas Advanced Research Program Number 4659 and by the National Science Foundation under Grant ECS-8217668.

W. K. Tsai, G. Huang, and J. K. Antonio are with the Department of Electrical Engineering, Texas A&M University, College Station, TX 77843-3128.

W. T. Tsai is with the Department of Computer Science, University of Minnesota, Minneapolis, MN 55455.

IEEE Log Number 8824237.

[1] In a related study, we have found a competitive algorithm called textured decomposition algorithm [25]. The companion textured decomposition algorithm for the nonlinear optimal load flow problem [26], [27] also shows tremendous speed advantage over classical algorithms.

Lipschitz continuously differentiable. To understand the (BP) we need the following fundamental lemma [17].

Lemma 1: If x^* is a local minimizing point of the (BP), then it must satisfy the following condition: for all $j = 1, \cdots, N$,

$$x_j^* > 0 \Rightarrow \frac{\partial J}{\partial x_j}(x^*) = 0,$$

$$x_j^* = 0 \Rightarrow \frac{\partial J}{\partial x_j}(x^*) \geq 0.$$

Thus, a vector x^* satisfying the condition of Lemma 1 is said to be a critical point of the (BP). We now introduce the Goldstein–Levitin–Polyak gradient projection update for the (BP). For $x \in \Re^N$, $y = [x]^+$ is defined by

$$y_j = \max(0, x_j), \qquad \forall j = 1, 2, \cdots, N.$$

The update equation for a serial gradient projection is given by

$$x(t+1) = [x(t) - \alpha(t)\nabla J(x(t))]^+$$

where $\alpha(t)$[2] is the stepsize at iteration t. It can be simply proven that if a small enough constant stepsize or a modified Armijo stepsize rule [14] is used, the above gradient projection algorithm converges to a local minimizing point of the (BP).

B. The Distributed Computation and Communication Model

The model presented below is a variation of the specialization case of the general distributed asynchronous computation model developed by Tsitsiklis *et al.* [16], which is a generalization of the earlier model by Bertsekas [18]. Our variation is a simplified model because the notations of the IAD method are already complicated, and a more general model will hinder the understanding of the essence of the method. We call the following computation model a *partially asynchronous* model as opposed to the *totally asynchronous* model by Bertsekas [18]. It can be described by the following assumptions.

1) The total number of events is countable and can be totally ordered. Thus, the system can be thought of operating in discrete time, i.e., $t \in \{0, 1, 2, \cdots\}$. Although a global clock is assumed, each processor does not need to have access to this global clock.

2) A connected network in which each node represents a processor is given. Each processor can communicate with every other processor via the network. The total number of processors assumed is l.

3) Let us decompose the space $\Re^N$ into a Cartesian product of l subspaces, $\Re^N = \Re^{N_1} \times \cdots \times \Re^{N_l}$, $\Sigma_j^l N_j = N$, and write any $x \in \Re^N$ in block vector form, $x = [x_j]_{j=1,\cdots,l}$. Each processor i is responsible for the variable (or a subvector) x_i of the total state variable vector x. An estimate of the state is stored at each processor and for processor i, the copy of the state estimate at time t is denoted by $x^i(t)$, $x^i(t) = [x_j^i(t)]_{j=1,\cdots,l}$.

4) Each processor i updates $x_i^i(t)$ at time t if and only if $t \in T^i$, where T^i is the set of times that processor i performs an update. The update is of the form

$$x_i^i(t+1) = x_i(t) + \alpha_i(t)s_i(t)$$

where $\alpha_i(t)$ is a positive stepsize. If $t \notin T^i$, we set $s_i(t) = 0$.

5) Let T_j^i denote the set of times that processor i receives an updated value of x_j^i from processor j. Suppose $t \in T_j^i$, then $x_j^i(t)$ is made equal to the newly received x_j^i. Since all messages arrive after some delay, we assume

$$x_j^i(t) = x_j^j(\tau_j^i(t))$$

where $\tau_j^i(t) \leq t$ represents the time that x_j^i was last updated at processor j before it was sent to processor i.

6) For every processor i, if $t \notin T^i$ and $t \notin T_j^i$ for every $j \neq i$, then

$$x^i(t+1) = x^i(t),$$

i.e., the state estimate does not change if no new update is performed and no new message is received.

7) For all i, j, $i \neq j$, T^i and T_j^i are infinite sets, i.e., the processors never stop communicating and computing. Such assumption is obviously needed for a distributed iterative algorithm to converge.

The following synchronization assumptions are needed for the partially asynchronous model. We assume that there is a positive number B such that the following assumptions are true.

Bounded Communication Delay (P1): For all $i \neq j$, $t - \tau_j^i(t) \leq B$, i.e., the communication delays are uniformly bounded by B.

Partially Asynchronous Communication (P2): For any $i \neq j$, the consecutive elements of T_j^i differ less than B.

Partially Asynchronous Computation (P3): For any i, the consecutive elements of T^i differ less than B.

From assumptions (P1) and (P2), for any $i \neq j$ and t, we have

$$t - 2B \leq \tau_j^i(t) \leq t - 1.$$

Thus, the estimates of the state at two different processors are at most $2B$ time units out of date. From assumptions (P_1)–(P_3) we can simply prove the following agreement condition.

Agreement Condition (AC): For all $i = 1, 2, \cdots, l$, $j = 1, \cdots, N$, all $t \geq 0$

$$|x_j^i(t) - x_j^j(t)| = |x_j^j(\tau_j^i(t)) - x_j^j(t)| \leq \alpha \sum_{n=t-2B}^{t-1} |s_j(n)|.$$

The assumptions (P1)–(P3) are necessary, as one can construct counterexamples [13], [24] showing that the algorithm fails to converge if any one of the assumptions is violated. The agreement condition is a natural consequence of (P1)–(P3). It basically says that, if no computation update is performed, the local states at each processor will agree after $2B$ units of time. Such agreement is necessary for the proposed algorithm to converge. In (AC), we assume that $\tau_i^i(t) = t$ for all i and t for convenience; the interpretation is that processor i has $x_i^i(t)$ in its memory for all t.

III. Aggregation/Disaggregation Methods

A. Completeness of A/D Matrices

We follow the notation used in [4]. Let $p < N$ be an integer smaller than N. A $p \times N$ matrix R is referred to as a restriction matrix if the aggregate vector $\hat{x}$ is obtained from the unaggregated vector x via $\hat{x} = Rx$. The matrix R can be interpreted as the operator which aggregates the vector x; the dimension of $\hat{x}$, p is smaller (typically much smaller) than N, the dimension of x. An $N \times p$ matrix P is referred to as a prolongation matrix if the disaggregated vector $\tilde{x}$ is obtained from the aggregated vector $\hat{x}$ via $\tilde{x} = P\hat{x}$. The matrix P can be interpreted as the operator which prolongs (or disaggregates) the restricted vector $\hat{x}$ into a full vector. The matrix PR is denoted by Π; if $RP = I_p$ (which is usually the case in an IAD algorithm), the identity matrix in $\Re^p$, we also have $\Pi^2 = \Pi$, i.e., Π is a projection matrix.

There are many possible A/D (aggregation/disaggregation) methods for the (BP); we shall restrict ourselves to the most natural case. Before we state the A/D method, we introduce the concept of complete A/D sets.

A set of m pairs of restriction–prolongation matrices $\{R^k, P^k\}_{k=1,\cdots,m}$ is said to be complete if they can be put into the

[2] More generally, $\alpha(t)$ can be considered as the product of a stepsize and a scaling factor which is used to accelerate the convergence. As we focus on the convergence in this paper, we will ignore scaling in our analysis.

following form (reindex if necessary):

$$P^1=\begin{bmatrix} I^1 & \\ & \bar{L}^1 \end{bmatrix},\ R^1=\begin{bmatrix} I^1 & \\ & L^1 \end{bmatrix},$$

$$P^2=\begin{bmatrix} \bar{U}^2 & & \\ & I^2 & \\ & & \bar{L}^2 \end{bmatrix},\ R^2=\begin{bmatrix} U^2 & & \\ & I^2 & \\ & & L^2 \end{bmatrix},$$

$$\vdots \qquad\qquad \vdots$$

$$P^m=\begin{bmatrix} \bar{U}^m & \\ & I^m \end{bmatrix},\ R^m=\begin{bmatrix} U^m & \\ & I^m \end{bmatrix}$$

where U^k, L^k, $\bar{U}^k$, $\bar{L}^k$, are arbitrary matrices of proper dimensions and I^i are identity matrices such that

$$\begin{bmatrix} I^1 & & & \\ & I^2 & & \\ & & \ddots & \\ & & & I^m \end{bmatrix}=I_N$$

I_N being the identity matrix in the space $\Re^N$. The intuition behind this concept of completeness is that, in such an A/D method, each variable always has a corresponding A/D step at which it is not aggregated, i.e., free to vary on its own. Thus, the m A/D steps together provide N degrees of freedom for the algorithm. Such freedom is in general necessary to achieve optimality. We now state the proposed set of restriction–prolongation matrices for the (BP). Let $x \in \Re^N$, then the set $\{R^k\}_{k=1,2,\cdots,m}$ is as follows.

$$R^k=\begin{bmatrix} \underbrace{1\cdots 1}_{N_1^k} & & & \\ & \underbrace{1\cdots 1}_{N_2^k} & & \\ & & \ddots & \\ & & & \underbrace{1\cdots 1}_{N_{n^k}^k} \end{bmatrix}, \tag{1}$$

$$P^k(x)=\begin{bmatrix} \frac{x_1}{z_1} & & & \\ & \frac{x_2}{z_2} & & \\ & & \ddots & \\ & & & \frac{x_{n^k}}{z_{n^k}} \end{bmatrix} \tag{2}$$

where $z_j = \Sigma_p x_{j,p}$, $x_{j,p}$ is the pth component of the subvector x_j, and x is written in the block vector form $x = [x_j]_{j=1,\cdots,n^k}$, assuming $\Sigma_p x_{j,p} > 0$ for all j. Note that for the above class of prolongation matrices, each $P^k(x)$ is a block diagonal matrix. To suit our distributed computation model, we assume that the set of subvectors $\{x_j\}_{j=1,\cdots,n^k}$, for each k can be distributed among l processors (i.e., $n^k \geq l$), and we can write $x = [x_i]_{i=1,\cdots,l}$, where each x_i is a vector formed by cascading several x_j's. With this partition of x, the matrix $P^k(x)$ can be put into a block-diagonal form

$$P^k(x)=\text{diag}\,\{P_i^k(x)\}. \tag{3}$$

As a convention, when a processor uses a certain pair of $\{R^k, P^k(x)\}$, it is said to use the aggregation policy k at x. For simplicity, we call these prolongation and restriction matrices A/D matrices. We shall assume that all the set of m A/D matrices mentioned in the subsequent development satisfy the completeness condition and have the form (1)–(3).

B. The IAD-GP Algorithm

There are numerous ways of incorporating the different A/D methods in a GP algorithm. There are two general approaches, one advocated by the Soviet researchers and the other by Miranker and his colleagues. In the first approach, A/D steps are used at each iteration. In the second approach, an A/D step is applied intermittently among the iterations; A/D steps are used mainly for acceleration purposes. Our IAD-GP program coded for optimal routing follows the second approach.

To highlight the essence of the IAD-GP algorithm we only state a synchronous version which encompasses both approaches.

Assume that the set of A/D matrices is a complete set. Then for $t = 0, m, 2m, \cdots$, such that for all $k = 0, 1, 2, \cdots, m-1$,

$$\hat{x}(t+k)=R^{k+1}(x(t+k))x(t+k). \quad \text{(a : Restriction)}$$

$$\hat{x}(t+k+1)=[\hat{x}(t+k)-\alpha\nabla \hat{J}^{k+1}_{x(t+k)}(\hat{x}(t+k))]^+. \quad \text{(b : Descent Update)}$$

$$x(t+k+1)=P^{k+1}(x(t+k))\hat{x}(t+k+1) \quad \text{(c : Prolongation)}$$

where $\hat{J}^k_x(y) = J(P^k(x)y)$, $y \in \Re^{n^k}$ is the aggregated cost function using the prolongation matrix $P^k(x)$.

Note that if $R^k = I_N$, $P^k(x) = I_N$, then the IAD-GP update with policy k at x is the same as the usual GP update, i.e., all the variables are not aggregated. Thus, the above serial algorithm includes both approaches by the Soviet researchers and Miranker and his colleagues.

IV. The Convergence of the Distributed IAD-GP Algorithm

A. Some Preliminaries

As our focus in this paper is not on the convergence rate analysis, we assume in this paper that the stepsize is the same constant for all processors. Actually, in a distributed implementation, a small constant stepsize is safer than a distributed version of a more elaborate stepsize rule. One reason for the difficulty associated with an elaborate stepsize rule is that in a distributed computational environment, most processors do not have an accurate estimate of the global state vector and an unexpected large stepsize resulting from such a rule can be quite damaging. Before stating the convergence result, we state some preliminary facts.

Let $\langle\cdot\rangle$ and $\|\cdot\|$ denote the usual inner product and the usual Euclidean norm or induced Euclidean norm in appropriate Euclidean spaces, and superscript T denote transpose.

Lemma 2: For any $x \in \Re^N$, $1 \leq k \leq m$, such that $P^k(x)$ is well-defined $x = [x_j]_{j=1,\cdots,n^k}$ then the following is true.

a) $P^k(x)R^k x = x$.

b) $\nabla \hat{f}^k_x(y) = [P^k(x)]^T \nabla f(y)$, for all $y \in \Re^N$.

c) Suppose $\hat{x}^k(\alpha) = (\hat{x}^k + \alpha \hat{d}^k)^+$, $x^k(\alpha) = P^k(x)\hat{x}^k(\alpha)$, $\hat{x}^k = R^k x$, for some scalar α, and some vector $\hat{d}^k$, then $x^k(\alpha) = (x + \alpha P^k(x)\hat{d}^k)^+$.

d) $\|P^k(x)\| \leq 1$ and $\|P_i^k(x)\| \leq 1$ for all $i = 1, \cdots, l$.

Proof: a) is trivially true. b) follows by the chain rule in differentiation. c) Consider any pth component of $x_j^k(\alpha)$: $x^k_{j,p}(\alpha)$. Then

$$x^k_{j,p}(\alpha)=\frac{x_{j,p}}{\sum_p x^k_{j,p}}\hat{x}_{j,p}(\alpha)$$

$$=\frac{x_{j,p}}{\sum_p x_{j,p}}\left[\sum_p x_{j,p}+\alpha\hat{d}^k_{j,p}\right]^+$$

$$=\left[x_{j,p}+\alpha\frac{x_{j,p}}{\sum_p x_{j,p}}\hat{d}^k_{j,p}\right]^+.$$

d) For any real matrix A, $\|A\|^2 = \max\{\lambda: \lambda \text{ is an eigenvalue of } A^TA\}$. Note that

$$[P^k(x)]^TP^k(x) = \begin{bmatrix} \frac{x_1^Tx_1}{z_1^2} & & & \\ & \frac{x_2^Tx_2}{z_2^2} & & \\ & & \ddots & \\ & & & \frac{x_{m_k}^Tx_{m_k}}{z_{m_k}^2} \end{bmatrix}.$$

By the fact that $a^2 + b^2 \le (a+b)^2$ for any scalars $a \ge 0$, $b \ge 0$, we have $x_j^Tx_j \le (\Sigma_k x_{j,k})^2$. Thus, $\|P^k(x)\|^2 = \max_j \{x_j^Tx_j/(\Sigma_k x_{j,k})^2\} \le 1$. Similarly, for each $1 \le i \le l$, $\|P_i^k(x)\| \le 1$. Q.E.D.

From Lemma 2c), the IAD–GP updates a)–c) can be written on the original space equivalently: for some k

$$x(t+1) = [x(t) - \alpha P^k(x(t))[P^k(x(t))]^T\nabla J(x(t))]^+. \quad (4)$$

At each update, the local state at each processor makes at most $O(\alpha)$ change, thus the difference between a partially asynchronous algorithm, and the serial algorithm is on the order of $2B\alpha$. Now we introduce the central idea of the proof method. The idea is to keep track of an important global state vector. The most natural global state vector consists of $\{x_i^i(t)\}_{i=1,\cdots,l}$, and the vector of all the local steps $\{s_i(t)\}_{i=1,\cdots,l}$, are also important

$$y(t) = \begin{bmatrix} x_1^1(t) \\ \vdots \\ x_l^l(t) \end{bmatrix}, \quad s(t) = \begin{bmatrix} s_1(t) \\ \vdots \\ s_l(t) \end{bmatrix}.$$

With the above definition, we have

$$y(t+1) = y(t) + \alpha s(t). \quad (5)$$

Now from the agreement condition (AC) we have $x^i(t) \approx y(t)$ for all i. Thus, if (5) is a convergent serial algorithm, then the partially asynchronous algorithm will also converge.

B. The Convergence Result

For simplicity, we ignore the time argument t in many of the subsequent developments. We need the following notations. For any $k = 1, 2, \cdots, m$,

$$Z^k(x) = P^k(x)[P^k(x)]^T, \quad \hat{x}^k = R^k(x)x,$$

$$x^k(\alpha) = [x - \alpha Z^k(x)\nabla J(x)]^+, \quad (6)$$

$$\hat{x}^k(\alpha) = [\hat{x}^k - \alpha(P^k(x))^T\nabla J(x)]^+. \quad (7)$$

From Lemma 2b) and c), (6) and (7) are the updates, respectively, in the original space and aggregate space if all the processors update simultaneously and they together implement the IAD–GP update using $\{R^k, P^k(x)\}$. From (6) and (7) it is readily seen that $x^k(\alpha) = P^k(x)\hat{x}^k(\alpha)$ and $P^k(x)R^kx = x$. The next theorem (a proof is provided in the Appendix) gives a necessary condition for optimality of the (BP) in the context of the IAD–GP algorithm.

Theorem 3: If the set of A/D matrices $\{R^k, P^k\}$ is complete, then the following is true.

a) $x \in \mathcal{R}^N$ is a critical point of the (BP) if and only if

$$x^k(\alpha) = x, \ \forall \alpha \ge 0, \qquad \forall k = 1, 2, \cdots, m.$$

b) For any $k = 1, 2, \cdots, m$, let $\hat{s}_j^k(\alpha; \hat{x}) = 1/\alpha[\hat{x}_j^k(\alpha) - \hat{x}_j^k]$, where $\hat{x}_j^k(\alpha)$ and $\hat{x}_j^k$ are the jth components of $\hat{x}^k(\alpha)$ and $\hat{x}^k$, respectively, for all $j = 1, 2, \cdots, n^k$, then

$$\frac{\partial \hat{J}^k}{\partial x_j}(\hat{x})\hat{s}_j^k(\alpha; \hat{x}) \le -|\hat{s}_j^k(\alpha; \hat{x})|^2. \quad (8)$$

We now state the main convergence theorem; the proof can be found in the Appendix.

Theorem 4: With the IAD–GP algorithm and the distributed computation model described in Sections II and III, if the set of A/D matrices $\{R^k, P^k\}$ is complete, and $J(x) \to \infty$ as $\|x\| \to \infty$, then there exists a $\bar{\alpha} > 0$ such that for all $\alpha \in (0, \bar{\alpha}]$: a) the sequence $\{J(x^i(t))\}$ converges for all i; b) for all i, $\lim_{t\to\infty} \|x^i(t) - y(t)\| = 0$; c) if y^* is a limit point of the sequence $\{y(t)\}$, then y^* is a critical point of the (BP).

According to (4), the IAD–GP algorithm appears to be similar to the scaled GP algorithm in Tsitsiklis and Bertsekas [13]. Thus, the proofs of Theorems 3 and 4 generally follow the structure of the convergence proof in [13]. In addition, we would like to comment on the difference between our convergence results and the results obtained by [13]. First, both the optimality result (Theorem 3) and the convergence result (Theorem 4) requires extra work to incorporate A/D procedures in the algorithm. Second, the IAD–GP update (4) may appear to be similar to the GP update of [13] with the scaling matrix equal to $P^k(x)[P^k(x)]^T$. However, [13] requires that the scaling matrix be uniformly positive definite (cf. [13, eq. (3.10)]), while it is possible that our $P^k(x)[P^k(x)]^T$ matrix may be only positive semidefinite; for example, the case where the vector x contains zero components [cf. (2)], and this situation almost always happens in the optimal routing problem to be considered next. The reason that we can do away with the uniformly positive definite condition is that we have chosen a specific class of scaling matrices whose special properties enables the elimination of the condition. In addition, in [13], the box-constrained version of the GP update needs the scaling matrix to be diagonal, while our $P^k(x)[P^k(x)]^T$ can be block diagonal. This may be due to the fact that we are assuming a slightly simpler distributed model than [13]; for example, we do not model the difference between desired[3] variables and actual variables (cf. [13, eq. (3.11)]). Although not proven yet, we expect the IAD–GP algorithm to converge even in the model described by [13]; again, the reason is that our scaling matrices are quite special.

V. Optimal Routing in Data Network

A. Problem Formulation

The optimal routing problem (ORP) in data networks can be formulated as a multicommodity flow problem as follows. Suppose that a directed graph $G = (\mathfrak{N}, \mathfrak{L})$, where $\mathfrak{N}$ is the set of nodes, and $\mathfrak{L}$ is the set of directed links, is given. A set of origin–destination (OD) pairs W, is given, and for each OD pair $w \in W$, the traffic demand r_w is given also. The purpose of (ORP) is to find a set of path flows $[x_w]_{w\in W}$ which satisfies the traffic demand

$$\sum_{p\in P_w} x_{w,p} = r_w, \qquad \forall w \in W \quad (9)$$

where P_w is a given set of paths for an OD pair w, and $x_{w,p}$ is the traffic flow on path p for w. The traffic balance equations which state that the total traffic flow on any directed link is the sum of all the path flows for all the paths using that directed link must also be satisfied. Let F_{ij} be the traffic flow on link $(i, j) \in \mathfrak{L}$, then the traffic balance equations can be written as follows:

$$F_{ij} = \sum_{w\in W} \sum_{(i,j)\in p, p\in P_w} x_{w,p}, \qquad \forall (i,j) \in \mathfrak{L} \quad (10)$$

[3] In a realistic virtual circuit network, the path flow variables cannot not generally be adjusted by the routing controllers instantly, thus the model in [13] differentiates desired flows from actual flows. However, to add this complication to our model, our already complex notation will become too burdensome to read; thus, we choose to avoid this complication.

or in matrix from $F = Hx$, where $((i,j),p)$th entry of H is one if (i,j) is in path p otherwise zero. Finally, all the traffic flows $x = [x_w]_{w\in W}$ must be nonnegative

$$x_{w,p} \geq 0, \qquad \forall p \in P_w, \forall w \in W. \tag{11}$$

Any path flow vector satisfying (9) and (11) is said to be admissible and we let X denote the set of admissible path flow vectors. The purpose of the (ORP) is to find a set of path flows that satisfies the constraints (9)–(11) and minimizes the following separable cost:

$$\bar{D}(F) = \sum_{(i,j)\in\mathcal{L}} \bar{D}_{ij}(F_{ij}) \tag{12}$$

where $\bar{D}_{ij}(\cdot)$ is usually assumed to be convex nondecreasing and Lipschitz continuously differentiable. The total cost $\bar{D}(F)$ is usually a measure of the congestion level in the network; a common cost is the average delay for each packet to traverse the network.

Now, the constraint (10) can be eliminated by substituting $F = Hx$ in (17): $D(x) = \bar{D}(Hx)$. $D(x)$ inherits all the smoothness and convexity properties of $\bar{D}(F)$. The optimality condition for the above (ORP) can be easily derived using Lemma 1 and is stated below (see also [28])

$$x_{w,p} = 0 \Rightarrow d_{w,p}(x) - d_{\bar{p}_w}(x) = 0,$$

$$x_{w,p} > 0 \Rightarrow d_{w,p}(x) - d_{\bar{p}_w}(x) \geq 0$$

where $d_{w,p}$ denote $\partial D(x)/\partial x_{w,p}$, and $\bar{p}_w$ is the shortest path according to the first derivative length

$$d_{\bar{p}_w}(x) = \min_{q\in P_w} \{d_{w,q}(x)\}.$$

B. The Aggregation and Disaggregation Procedure

We will now show how to aggregate and disaggregate without actually constructing A/D matrices. We also derive the formulas for partial derivative of the aggregate cost functions. We assume that the A/D matrices described by (1) and (2) are used. From (2) and (3), it is clear that first partial derivatives are the key information in most gradient projection methods for solving path-flow formulated multicommodity problems. In addition, second partial derivatives are usually used to accelerate the convergence [15]. Following the usual nonlinear programming approach, consider the problem of minimizing the second-order approximation of the cost at each iteration, which is a quadratic programming problem (QP) of the form

$$\min_{x(\alpha)\in X} \left\{ \sum_{p\in P} \frac{\partial D(x)}{\partial x_p}(x_p(\alpha) - x_p) + \frac{1}{2\alpha} \sum_{p\in P} \sum_{q\in P} \frac{\partial^2 D(x)}{\partial x_p \partial x_q} (x_p(\alpha) - x_p)(x_q(\alpha) - x_q) \right\} \tag{13}$$

where P is the set of all admissible paths in the network, x_p is the flow in path p, x is a vector defined by $x = [x_p]_{p\in P}$, and X is the set of all admissible path flows. For convenience, we define $d_p(x) \triangleq \partial D(x)/\partial x_p$ and $d_{pq}(x) \triangleq \partial^2 D(x)/\partial x_p \partial x_q$.

Next, according to aggregation policy k, we partition P into a collection $\hat{P}^k$ of mutually disjoint groups (or classes) of paths

$$\hat{P}^k = \{\hat{p}^k : \cup \hat{p}^k = P \text{ and } \hat{p}^k \text{ are mutually disjoint}\},$$

$$P = \{p : \exists \text{ exactly one } \hat{p}^k \in \hat{P}^k \text{ such that } p \in \hat{p}^k\}.$$

We now pose the following aggregate (QP) or (AQP) which is basically a restricted version of the (QP).

$$\min \left\{ \sum_{\hat{p}^k\in\hat{P}^k} \sum_{p\in\hat{p}^k} d_p(x)(x_p(\alpha) - x_p) + \frac{1}{2\alpha} \sum_{\hat{p}^k\in\hat{P}^k} \sum_{\hat{q}^k\in\hat{P}^k} \sum_{p\in\hat{p}^k} \sum_{q\in\hat{q}^k} d_{pq}(x)(x_p(\alpha) - x_p)(x_q(\alpha) - x_q) \right\}$$

such that

$$x_p(\alpha) = u_{\hat{p}^k} x_p \qquad \text{for all } p \in \hat{p}^k \text{ and all } \hat{p}^k \in \hat{P}^k,\ x(\alpha) \in X \tag{14}$$

where

$$u_{\hat{p}^k} > 0 \qquad \text{for all } \hat{p}^k \in \hat{P}^k.$$

If we substitute the x_p variables with $x_{\hat{p}^k}$ given by

$$x_{\hat{p}^k}(\alpha) = \sum_{p\in\hat{p}^k} x_p(\alpha) = u_{\hat{p}^k} \sum_{p\in\hat{p}^k} x_p \tag{15}$$

then we can express the (AQP) as

$$\min \left\{ \sum_{\hat{p}^k\in\hat{P}^k} (x_{\hat{p}^k}(\alpha) - x_{\hat{p}^k}) d_{\hat{p}^k}(x) + \frac{1}{2\alpha} \sum_{\hat{p}^k\in\hat{P}^k} \sum_{\hat{q}^k\in\hat{P}^k} d_{\hat{p}^k\hat{q}^k}(x)(x_{\hat{p}^k}(\alpha) - x_{\hat{p}}))(x_{\hat{q}^k}(\alpha) - x_{\hat{q}^k}) \right\} \tag{16}$$

subject to x satisfying (14), and where

$$x_{\hat{p}^k} = \sum_{p\in\hat{p}^k} x_p, \tag{17}$$

$$d_{\hat{p}^k}(x) = \frac{\sum_{p\in\hat{p}} x_p d_p(x)}{x_{\hat{p}^k}}, \tag{18}$$

$$d_{\hat{p}^k\hat{q}^k}(x) = \frac{\sum_{p\in\hat{p}^k} \sum_{q\in\hat{q}^k} d_{pq} x_p x_q}{x_{\hat{p}^k} x_{\hat{q}^k}}. \tag{19}$$

Now the natural interpretation of these new variables is as follows. $x_{\hat{p}^k}$ is the aggregate path flow for path $\hat{p}^k$. $d_{\hat{p}^k}(x)$ is the aggregate first partial derivative for path $\hat{p}^k$. And finally, $u_{\hat{p}^k}$ is the ratio of change in aggregate path flow from $x_{\hat{p}^k}$ to $x_{\hat{p}^k}(\alpha)$. And $d_{\hat{p}^k\hat{q}^k}(x)$ can be interpreted as the aggregate partial with respect to $\hat{p}^k$ and $\hat{q}^k$.

C. Convergence Analysis

Using the above (AQP) formulation, we can dynamically transform the (ORP) into a (BP). To do this, we have to make an additional assumption on the class of admissible A/D matrices and formulate the corresponding set of aggregate (ORP)'s. We require that if an aggregate path consists of detailed paths from a group of OD pairs, say group A, then for any other aggregate path which contains a path from one OD pair from group A, this other aggregate path must exactly contain paths from each OD pair in group A. This assumption is mathematically convenient for us to transform an (ORP) into the following aggregate (ORP), which we call (AORP-k); in practice, this assumption can be relaxed. Following this additional assumption, for each aggregate policy k, the overall OD pairs can be grouped into a set of disjoint

aggregate OD pairs

$$\hat{W}^k = \{\hat{w}^k : \cup \hat{w}^k = W \text{ and } \hat{w}^k \text{ are mutually disjoint}\}$$

$$W = \{w : \exists \text{ exactly one } \hat{w}^k \in \hat{W}^k \text{ such that } w \in \hat{w}^k\}.$$

For convenience, we also define, for each $\hat{w}^k \in \hat{W}^k$,

$$\hat{P}^k_{\hat{w}^k} = \{\hat{p}^k \in \hat{P}^k : \text{if } p \in \hat{p}^k, \text{ then } \exists w \in \hat{w}^k \text{ s.t. } p \in P_w\}.$$

The (AORP-k) is stated as follows:

$$\min D(x)$$

$$\text{s.t.} \sum_{\hat{p}^k \in \hat{P}^k_{\hat{w}^k}} x_{\hat{p}^k} = \sum_{w \in \hat{w}^k} r_w \triangleq r_{\hat{w}^k}, \quad \forall \hat{w}^k \in \hat{W}^k,$$

$$x_{\hat{p}^k} \geq 0, \qquad \forall \hat{p}^k \in \hat{P}^k_{\hat{w}^k}, \ \forall \hat{w}^k \in \hat{W}^k. \tag{20}$$

We are now ready to transform the above aggregate (ORP) into a (BP). The simplex constraint (20) can be eliminated by substitution. Let $d_{\hat{p}^k}(x) = \partial D/\partial x_{\hat{p}^k}(x)$ denote the first derivative of the cost with respect to aggregate path flow $x_{\hat{p}^k}$. Let $\hat{p}^{k,s}_{\hat{w}^k}$ be the shortest path according to the measure $\{d_{\hat{p}^k}(x)\}_{\hat{p}^k \in \hat{P}^k_{\hat{w}^k}}$ for the aggregate OD pair $\hat{w}^k$

$$d_{\hat{p}^{k,s}_{\hat{w}^k}}(x) = \min_{\hat{p}^k \in \hat{P}^k_{\hat{w}^k}} \{d_{\hat{p}^k}(x)\}. \tag{21}$$

Substituting $x_{\hat{p}^{k,s}_{\hat{w}^k}} = r_{\hat{w}^k} - \Sigma_{\hat{p}^k \in \hat{P}^k_{\hat{w}^k}, \hat{p}^k \neq \hat{p}^{k,s}_{\hat{w}^k}} x_{\hat{p}^k}$ for each $\hat{w}^k$ in $D(x)$, we get a new cost function $\tilde{D}^k(\tilde{x}^k)$ where $\tilde{x}^k = [[x_{\hat{p}^k}]_{\hat{p}^k \in \hat{P}^k_{\hat{w}^k}, \hat{p}^k \neq \hat{p}^{k,s}_{\hat{w}^k}}]_{\hat{w}^k \in \hat{W}^k}$. This substitution must be done at each iteration conceptually because the shortest paths $\{\hat{p}^{k,s}_{\hat{w}^k}\}_{\hat{w}^k \in \hat{W}^k}$ are varying in time. Thus, the new problem is now

$$\min \{\tilde{D}^k(\tilde{x}^k) : \tilde{x}^k \geq 0\}. \tag{BP-k}$$

Since

$$\frac{\partial \tilde{D}^k(\tilde{x}^k)}{\partial \tilde{x}^k_{\hat{p}^k}} = \frac{\partial D(x)}{\partial x_{\hat{p}^k}} + \frac{\partial D(x)}{\partial x_{\hat{p}^{k,s}_{\hat{w}^k}}} \frac{\partial x_{\hat{p}^{k,s}_{\hat{w}^k}}}{\partial x_{\hat{p}^k}}$$

we will denote $\tilde{d}_{\hat{p}^k}(x) = d_{\hat{p}^k}(x) - d_{\hat{p}^{k,s}_{\hat{w}^k}}(x)$, the first partial derivative of the cost $\tilde{D}^k(\tilde{x}^k)$ with respect to $\tilde{x}_{\hat{p}^k}$. Let also $\tilde{d}^k(x) = [[\tilde{d}_{\hat{p}^k}(x)]_{\hat{p}^k \in \hat{P}^k_{\hat{w}^k}, \hat{p}^k \neq \hat{p}^{k,s}_{\hat{w}^k}}]_{\hat{w}^k \in \hat{W}^k}$.

We will now show this dynamic change of variable together with the IAD–GP updates guarantee that the nonnegativity constraints for the aggregate shortest path flows after the updates are not violated. In terms of the new variables, the IAD–GP update is

$$\tilde{x}^k(\alpha) = [\tilde{x}^k - \alpha \tilde{d}^k(x)]^+. \tag{22}$$

Since $d_{\hat{p}^{k,s}_{\hat{w}^k}}(x) = \min_{\hat{p}^k \in \hat{P}^k_{\hat{w}^k}} \{d_{\hat{p}^k}(x)\}$ [from (21)], we have $\tilde{d}^k(x) \geq 0$ (the inequality is taken componentwise). Thus, $x_{\hat{p}^k}(\alpha) \leq x_{\hat{p}^k}$, for all $\hat{p}^k \in \hat{P}^k_{\hat{w}^k}, \hat{p}^k \neq \hat{p}^{k,s}_{\hat{w}^k}$, for all $\hat{w}^k \in \hat{W}^k$. Hence, for all $\hat{w}^k$, $x_{\hat{p}^{k,s}_{\hat{w}^k}}(\alpha)$ defined by

$$x_{\hat{p}^{k,s}_{\hat{w}^k}}(\alpha) = r_{\hat{w}^k} - \sum_{\hat{p}^k \in \hat{P}^k_{\hat{w}^k}, \hat{p}^k \neq \hat{p}^{k,s}_{\hat{w}^k}} x_{\hat{p}^k}(\alpha) \tag{23}$$

we have the property $x_{\hat{p}^{k,s}_{\hat{w}^k}}(\alpha) \geq x_{\hat{p}^{k,s}_{\hat{w}^k}} \geq 0$, and (11) is not violated for the aggregate shortest path flows.

To establish the fact that the dynamically transformed (AORP-k) can be solved by the iterative distributed IAD–GP algorithm, we need to show the counterparts of Theorems 3 and 4.

Theorem 3a: Suppose the set of A/D matrices $\{R^k, P^k\}$ satisfies the condition that for each OD pair w, there exists one pair of A/D matrices according to which every path in P_w is not aggregated (which implies that the set of A/D matrices is complete), then the following is true.

a) x is a critical point of (ORP) if and only if

$$\tilde{x}^k(\alpha) = \tilde{x}^k, \qquad \forall \alpha \geq 0, \ \forall k = 1, 2, \cdots, m.$$

b) For any $k = 1, 2, \cdots, m$, let $s_{\hat{p}^k}(\alpha; x) = 1/\alpha[\tilde{x}^k_{\hat{p}^k}(\alpha) - \tilde{x}^k_{\hat{p}^k}]$, where $\tilde{x}^k_{\hat{p}^k}(\alpha)$ and $\tilde{x}^k_{\hat{p}^k}$ are the $\hat{p}^k$th components of $\tilde{x}^k(\alpha)$ and $\tilde{x}^k$, respectively. Then for all $\hat{p}^k \in \hat{P}^k$, $\hat{p}^k \neq \hat{p}^{k,s}_{\hat{w}^k}$, for every $\hat{w}^k$ then

$$\tilde{d}_{\hat{p}^k}(x) s_{\hat{p}^k}(\alpha; x) \leq -|s_{\hat{p}^k}(\alpha; x)|^2. \tag{8a}$$

The main difference between Theorems 3 and 3a are the condition on the A/D matrices and the adaptation for the (ORP). It is easily seen that the special condition on the A/D matrices implies that the A/D matrices are complete which is the original condition in Theorem 3. The new condition is actually quite general and is very convenient in implementing the algorithm; the large class of aggregation methods presented in the next section satisfy this new condition. The proof of Theorem 3a and the proof of Theorem 3 differ in that Theorem 3 part a) states the equivalence between the fixed point condition and the optimality condition of the (BP), while Theorem 3a part a) states the equivalence between the fixed point condition and the optimality condition of the (ORP).

To apply the distributed IAD–GP algorithm to the (ORP), we impose a natural separation assumption on the assignment of the state variables to the processors.

Separation Assumption: Each processor i is responsible for all the path routing (flow) variables for a group $W^i \subset W$ of OD pairs. Thus, for aggregation policy k, for processor i, we can find the corresponding aggregate OD pairs $W^{k,i}$ formed from W^i.

The counterpart Theorem 4 is now stated below.

Theorem 4a: Under the same condition as in Theorem 3a plus the separation assumption and the assumption that $D(x) \to \infty$ as $\|x\| \to \infty$, there exists a $\bar{\alpha} > 0$ such that for all $\alpha \in (0, \bar{\alpha}]$: a) the sequence $\{D(x^i(t))\}$ converges for all i; b) for all i, $\lim_{t \to \infty} \|x^i(t) - y(t)\| = 0$; c) if y^* is a limit point of the sequence $\{D(t)\}$, then y^* is a critical point of the (ORP).

A proof of Theorem 4a can be made very similar to the proof of Theorem 4 by expressing all the terms in the Taylor series expansion (cf. (A8) in the Appendix) in terms of the original cost D instead of the transformed cost $\tilde{D}^k$. Our proof provided in the Appendix follows this approach.

VI. Aggregation Methods for Optimal Routing

Optimal routing in large interconnected data networks naturally lends itself to the IAD–GP algorithm. Large wide area networks are usually formed by interconnecting smaller networks (Fig. 1 shows such a network), and most networks are designed by using a hierarchical topological structure. These properties of large networks naturally fit the hierarchical multilevel IAD algorithms. A hierarchical structure is perhaps the best way to manage an existing large network, for example, large packet radio networks can have hundreds or thousands of mobile nodes. An example of a hierarchical organization of a large network is shown in Fig. 2. In fact, a hierarchical multilevel management structure has been suggested and implemented for packet radio networks [19]–[22]. In this section, we will show a general class of aggregation methods for the IAD–GP algorithm applied to optimal routing in large data networks.

We again restrict ourselves to a two-level hierarchy. The A/D method starts by partitioning the network into several level-1 clusters, and one level-2 cluster. First, we state the following two definitions.

- A *trunk line* is any link which directly connects two nodes belonging to distinct areas.
- The starting and ending nodes of a trunk line are called *gates*.

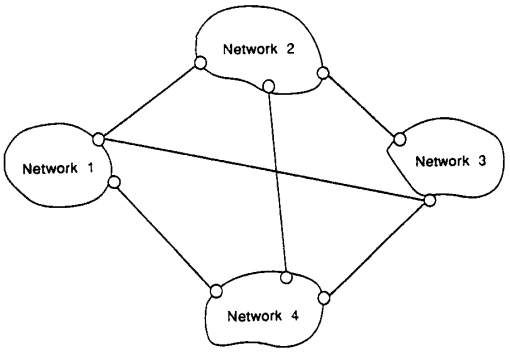


Fig. 1. Hierarchical network structure.

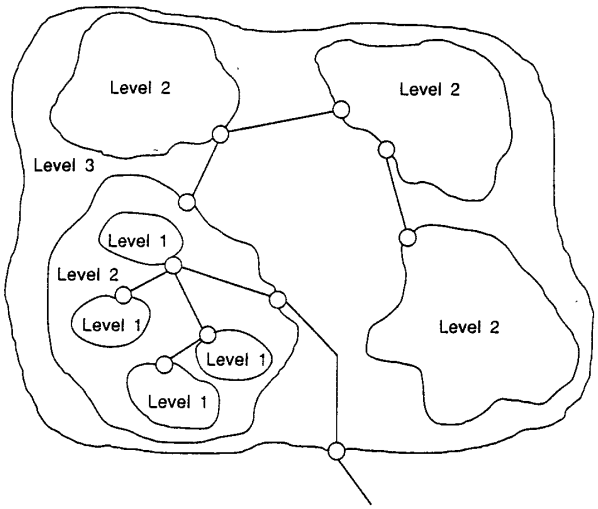


Fig. 2. Multilevel network structure.

Each level-1 cluster must have strictly fewer number of gates than its total number of nodes; a level-1 cluster with all its nodes as gates has unnecessary overheads. The level-2 cluster consists of all the trunk lines and all the gates. Using Fig. 1, as an example, Net 1, Net 2, Net 3, and Net 4 are all level-1 clusters, and the network encircled by the dotted line including all the gates and all the trunk lines is the level-2 cluster. We consider only the following four types of aggregation.

i) Down-Stream Aggregation: All the traffic streams (or paths) of the same OD pair, which originate at a node inside a level-1 cluster and end outside the cluster going through the same gate using the same route inside the cluster, are aggregated. A virtual node is created to represent the world (or the destination). The picture seen by the routing controller of an OD pair in this aggregation is illustrated in Fig. 3.

ii) Up-Stream Aggregation: This is the dual of i). All the traffic streams of the same OD pair, which originate outside a level-1 cluster and end at a node inside the cluster coming through the same gate using the same route outside the cluster, are aggregated. A virtual node is created to represent the world (or the origin). The picture seen by the routing controller at one of the gates in this aggregation is illustrated in Fig. 4.

iii) Mid-Stream Aggregation: All the traffic streams of the same OD pair, which originate outside a level-1 cluster and end outside the same cluster coming through the same gate, leaving through the same gate, using the same route inside the cluster, are aggregated. Two virtual nodes are created to represent, respectively, the origin and destination. The picture seen by the controller at one of gates is illustrated in Fig. 5.

iv) Level-2 Aggregation: For each level-1 cluster, a virtual node is created to represent all the origins (destinations) inside the cluster, and all the traffic streams going (coming) through the same gate are aggregated. The picture seen by a routing controller at any node of the level-2 cluster is illustrated in Fig. 6.

Although we only present 2-level aggregation methods here, the more general multilevel hierarchical aggregation methods can be similarly derived following the 2-level methods.

Note that the above description of aggregation methods provides the direct information to construct the corresponding restriction matrices. For example, the flow of those paths which are to be aggregated are summed to a single aggregate path flow. Also, in actual computation, we do not need to compute the restriction matrices; we really need to know what paths are aggregated to form various aggregate paths and then compute the sums. Similarly, we do not need to compute the prolongation matrices in actual implementation; we just need to know the nonzero entries of the matrices and perform the prolongation by simpler calculation (cf. Section V-B).

VII. Implementation Results

A. Basic Structure of the IAD-GP Program

We implemented a serial version of the proposed IAD-GP algorithm for a 52-node network. The program described here is actually a subroutine which is used in conjunction with the MULTIFLO program developed by Bertsekas, Gendron, and Tsai [23]. The A/D program is called by MULTIFLO after certain iterations. The overall goal of the A/D program is to speed up the rate of convergence in determining the optimal routing solution.

The program consists of three main parts.

Part 1: Aggregation of the original network.

Part 2: Solving the optimal aggregate routing problem approximately.

Part 3: Disaggregating the aggregate solution.

The first part basically creates a new aggregate network based on both the given partitioning (of levels) of the original network and the current flow patterns, provided by the main program.

The second part of the program solves approximately the new aggregate optimal routing problem (created by the first part) by using the same techniques of the main MULTIFLO program for a few iterations.

The third part of the program disaggregates the aggregate optimal path flow solution (determined by the second part) according to a weighted average of the original subpath flows. (An example of such a disaggregation calculation is given in Section V-B.) After all of the original path flows are updated, then all of the link flows of the original network are updated. Parts 1, 2, and 3 are done successively until a convergence criterion is met. The program then returns back to the main program where another iteration on the original network is done.

B. The Origin Area Destination Area (OADA) A/D Program

From the previous AQP derivation it is evident that a large class of A/D schemes is conceivable, since there are generally a multitude of choices for $\bar{P}$. We devised (and coded) a type of A/D scheme called OADA. In this OADA program, we only use the level-2 aggregation as described in Section VI. The utility of this OADA A/D algorithm is based on network structures characterized by relatively weak connections between areas. Roughly speaking, the connection between two areas is said to be relatively weak if the average delay for inter-area communication is large compared to the average delay for intra-area communication. As one might expect, this type of network structure is realistic when considering networks which span vast amounts of geographic area.

The fundamental requirement of OADA A/D is that networks

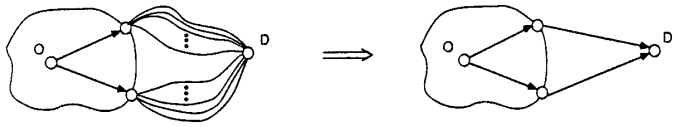


Fig. 3. Down-stream aggregation.

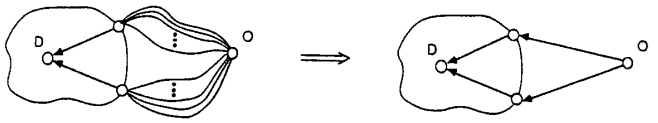


Fig. 4. Up-stream aggregation.

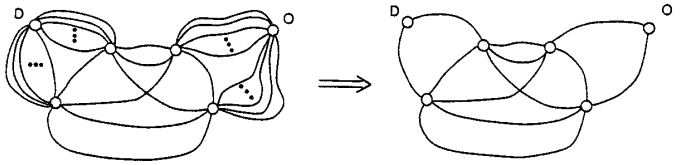


Fig. 5. Mid-stream aggregation.

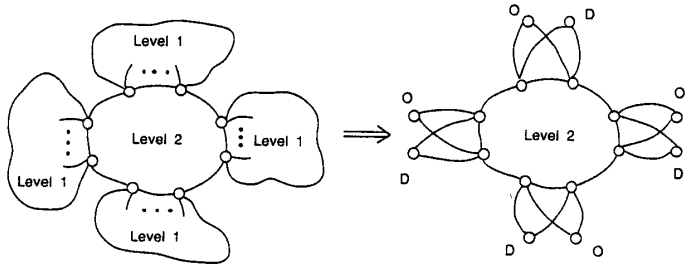


Fig. 6. Level-2 aggregation.

be partitioned into distinct nonoverlapping areas. That is, each node of the network belongs to one and only one area. Before describing how $\tilde{P}$ is determined, it is convenient to introduce the following terminology.

- For a particular OD pair, the area to which the origin belongs is called the *origin area*. Likewise, the area to which the destination belongs is called the *destination area*.
- The *active paths* for an OD pair are those paths (connecting origin to destination) which have position flow.

With the above terminology, we can now describe $\tilde{P}$. At iteration t, a typical element, say $\tilde{p} \in \tilde{P}$, would be a group of active paths satisfying the following two conditions.

1) The origin area of each active path is i and the destination area of each active path is j, where $i \neq j$.

2) All active paths leave origin area i through the same gate.

We apply the (OADA) A/D program in solving the optimal routing problem for the 52-node network depicted in Fig. 7. Every link in the network is bidirectional and has a capacity of 50 except the trunk lines which are rated at 80. Also, there are 85 OD pairs with an average[4] demand per OD pair of about 4.4 The cost function assumed is the average number of outstanding packets in the network, which is proportional to the average delay for each packet to traverse the network [15]. The initial routing is that every OD pair sends all its traffic via the shortest-hop path. We shall loosely call the cost average delay. A flowchart for the A/D algorithm is given in Fig. 8.

Fig. 9 shows the structure of the aggregate network. Note that the nodes labeled A1 and A4 represent the aggregate nodes for each area. Also, the links connecting A1 to 11, A1 to 9, A2 to 13, etc., are the aggregate origin links. The links connecting 11 to A1, 9 to A1, 13 to A2, etc., are the aggregate destination links. The links connecting 9 to 11, 11 to 9, 13 to 25, etc., are the aggregate transfer links. The maximum link utilization factor of the optimal routing is 0.929.

Table I shows a comparison in convergence rate (of the average delay) with and without the A/D subroutine. We ran 20 times for each method, the average measured CPU times and the standard deviation of the measured CPU times are given below. For the first case (with A/D) an A/D step was done after iterations 3 and 5. Also, the required CPU time for the 7 iterations with A/D has an average of 2.23 s, while the required CPU time for the 13 iterations necessary without A/D has an average of 3.35 s. The computer used in this simulation is a VAX 11/785.

The changes in the delay for both methods at each iteration are provided in Table II.

VIII. Conclusions

A new gradient projection algorithm for the box-constrained problem using the method of iterative aggregation and disaggregation is developed and shown to converge in a distributed asynchronous computation model. As an important application, the algorithm is applied (coded) to solve the optimal routing problem for large interconnected data communication networks. The new code shows a 35 percent improvement in serial computation as compared to the fast gradient projection code by Bertsekas *et al.* [23], for a 52-node network. The aggregation method used is the level-2 aggregation method described in Section VI.

There are several implications and extensions of this work.

[4] We choose the demands randomly.

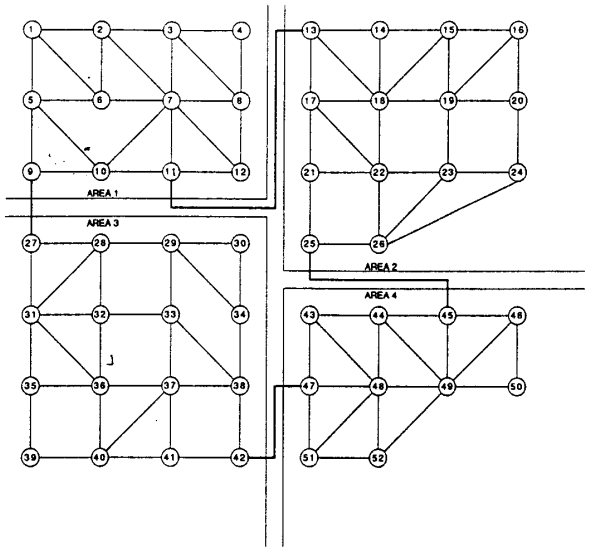


Fig. 7. The 52-node network.

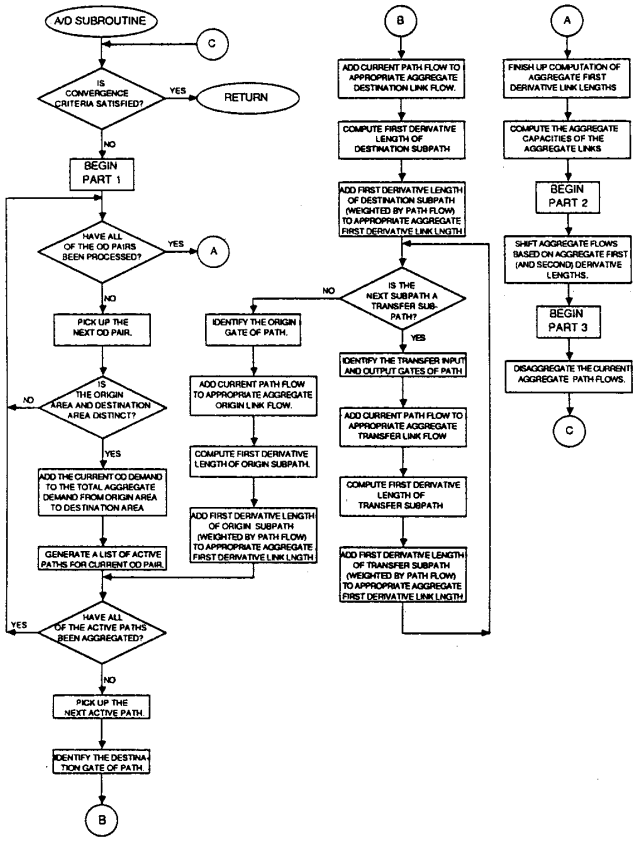


Fig. 8. Flowchart of the IAD–GP algorithm.

First, the iterative aggregation method has demonstrated its acceleration effects on an ordinary nonlinear programming algorithm. This is not surprising as the famous multigrid method [11] which is based on a similar aggregation/disaggregation idea is known to be the fastest algorithm for solving discretized elliptic partial differential equations. We expect that similar acceleration effects will result if the aggregation idea is applied to other nonlinear programming problems. In addition, if the IAD–GP algorithm is run in a synchronous computation environment, a modified Armijo rule patterned after Bertsekas [14] can be applied. The convergence proof is again very similar to the proof in this paper; only minor details need to be modified. One

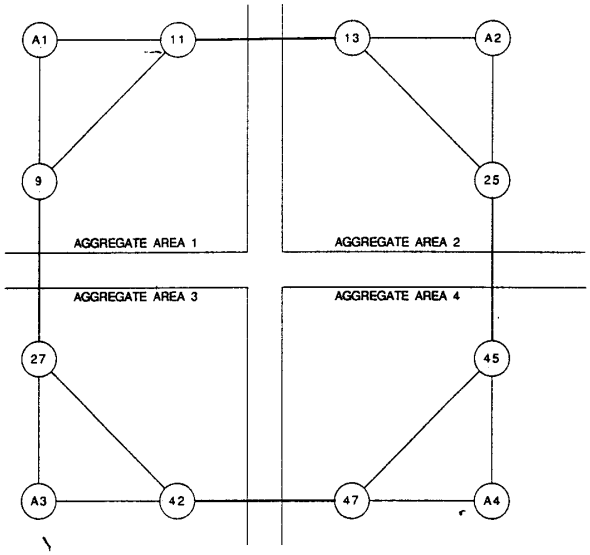


Fig. 9. The aggregate network.

TABLE I
CPU TIME COMPARISON

	with A/D	without A/D
Average CPU Time (sec.)	2.23	3.35
Standard Deviation of CPU Time (sec.)	0.084	0.16

TABLE II
COMPARISON IN AVERAGE DELAYS

Iteration Number	Average Delay (cost)	
	With A/D	Without A/D
1	1576113	1576113
2	1173	1173
3	284	284
4	205*	238
5	189	213
6	149*	197
7	147	188
8		179
9		170
10		161
11		154
12		150
13		147

*A/D step done before this iteration.

important next step is to analytically study the acceleration effect; this study is currently under way. From our experience with the algorithm for optimal routing, it appears that weak connections among clusters of strongly connected nodes play an important role in the acceleration effects.

APPENDIX

The purpose of this Appendix is to prove Theorems 3, 3a, 4, and 4a.

Proof of Theorem 3: a) Let $d^k(x) = [d_j^k(x)]_{j=1,\cdots,N} = Z^k(x)\nabla J(x)$, for $k = 1, 2, \cdots, m$. Assume that x is a critical point of the (BP). Then $x_j = 0$ implies that $\partial J/\partial x_j(x) \geq 0$, and $x_j > 0$ implies that $\partial J/\partial x_j(x) = 0$. These facts lead to $[P^k(x)]^T\nabla J(x) = 0$, for all k. Thus, $d^k(x) = 0$ for all k, $x^k(\alpha) = x^k$ for all $\alpha \geq 0$.

Conversely, assume that $x^k(\alpha) = x$ for all $\alpha \geq 0$, $k = 1, 2, \cdots, m$. From (6) we must have

$$d_j^k(x) = 0, \qquad \text{for all } x_j > 0, \tag{A1}$$

$$-d_j^k(x) \geq 0, \qquad \text{for all } x_j = 0. \tag{A2}$$

Since $\{R^k, P^k(x)\}_{k=1,\cdots,m}$ is a complete set of A/D matrices, we can from (A1) and (A2) derive that

$$x_j = 0 \Rightarrow \frac{\partial J}{\partial x_j}(x) = 0,$$

$$x_j > 0 \Rightarrow \frac{\partial J}{\partial x_j}(x) \geq 0.$$

b) Fix any $1 \leq i \leq m$. Denote

$$I_1^k(\hat{x}) = \left\{ 1 \leq j \leq n^k : \left(\hat{x}_j^k = 0 \text{ and } \frac{\partial \hat{J}^k}{\partial \hat{x}_j}(\hat{x}^k) \geq 0 \right) \text{ or } \left(\hat{x}_j^k > 0 \text{ and } \frac{\partial \hat{J}^k}{\partial \hat{x}_j}(\hat{x}^k) = 0 \right) \right\},$$

$$I_2^k(\hat{x}) = \left\{ 1 \leq j \leq n^k : \left(\hat{x}_j^k = 0 \text{ and } \frac{\partial \hat{J}^k}{\partial \hat{x}_j}(\hat{x}^k) < 0 \right) \text{ or } \left(\hat{x}_j^k > 0 \text{ and } \frac{\partial \hat{J}^k}{\partial \hat{x}_j}(\hat{x}^k) \neq 0 \right) \right\}.$$

Thus, $I_1^k(\hat{x})$ is the set of indexes that the corresponding components are not changed in the update equation (6); $I_2^k(\hat{x})$ is the set of indexes that the corresponding components are changed in the update equation (6).

Let $\hat{d}^k(\hat{x}) = [\hat{d}_j^k(\hat{x})]_{j=1,\cdots,n^k} = \nabla \hat{J}^k(\hat{x}^k) = (P^k(x))^T \nabla J(P^k(x)\hat{x})$. Let us now consider for any $j = 1, 2, \cdots, n^k$, the product

$$\frac{\partial \hat{J}^k}{\partial \hat{x}_j}(\hat{x})(\hat{x}_j^k(\alpha) - \hat{x}_j^k).$$

It is obvious that if $j \in I_1^k(\hat{x})$, then $\hat{x}_j^k(\alpha) - \hat{x}_j^k = 0$, and (8) is

satisfied. If $j \in I_2^k(X)$, we need to consider two cases: i) $\hat{x}_j^k(\alpha) > 0$; ii) $\hat{x}_j^k(\alpha) = 0$.

Suppose i) is true ($\hat{x}_j^k(\alpha) > 0$), then

$$\hat{d}_j^k(\hat{x})[\hat{x}_j^k(\alpha) - \hat{x}_j^k] = \hat{d}_j^k(\hat{x})[\hat{x}_j^k - \alpha\hat{d}_j^k(\hat{x}) - \hat{x}_j^k] = -\alpha(\hat{d}_j^k(\hat{x}))^2. \tag{A3}$$

Suppose ii) is true ($\hat{x}_j^k(\alpha) = 0$), then $\hat{x}_j^k - \alpha\hat{d}_j^k(\hat{x}) \le 0$. This case is possible only if $\hat{x}_j^k > 0$, $\hat{d}_j^k(x) > 0$. Thus, $\hat{x}_j^k \le \alpha\hat{d}_j^k(\hat{x})$. Now

$$\hat{d}_j^k(\hat{x})[\hat{x}_j^k(\alpha) - \hat{x}_j^k] = -\hat{d}_j^k(\hat{x})\hat{x}_j^k \le -\alpha[\hat{d}_j^k(\hat{x})]^2. \tag{A4}$$

Also

$$\alpha^2(\hat{d}_j^k(\hat{x}))^2 \ge (\hat{x}_j^k(\alpha) - \hat{x}_j^k)^2. \tag{A5}$$

Thus, (A3)-(A5) imply that for $j \in I_2^k(x)$,

$$\frac{1}{\alpha}\frac{\partial \hat{J}^k}{\partial \hat{x}_j}(\hat{x}^k)(\hat{x}_j^k(\alpha) - \hat{x}_j^k) \le -\left[\frac{\hat{x}_j^k(\alpha) - \hat{x}_j^k}{\alpha}\right]^2$$

which shows (8). Q.E.D.

Proof of Theorem 3a: a) Assume that x is a critical point of the (ORP). Then, by Lemma 1, $x_{w,p} = 0$ implies $d_{w,p}(x) - d_{p_w}(x) \ge 0$, and $x_{w,p} > 0$ implies $d_{w,p}(x) - d_{p_w}(x) = 0$ (see also [28]). Thus, $x_{\beta k} = 0$ implies $d_{\beta k}(x) - \bar{d}_{\beta \hat{w}^k s}(x) \ge 0$, and $\hat{x}_j^k > 0$ implies $d_{\beta k}(x) - \bar{d}_{\beta \hat{w}^k s}(x) = 0$ [from (18)]. These facts lead to $\bar{x}^k(\alpha) = \hat{x}^k$ for all $\alpha \ge 0$ and all k.

Conversely, assume that $\bar{x}^k(\alpha) = \bar{x}^k$ for all $\alpha \ge 0$, $k = 1, 2, \cdots, m$. From (33) we must have

$$\bar{d}_{\beta k}^k(x) = 0, \qquad \text{for all } \bar{x}_{\beta k}^k > 0, \tag{A6}$$

$$-\bar{d}_{\beta k}^k(x) \ge 0, \qquad \text{for all } \bar{x}_{\beta k}^k = 0. \tag{A7}$$

By the condition assumed for the set of A/D matrices, we can from (9a) and (10a) derive that

$$x_{w,p} = 0 \Rightarrow d_{w,p}(x) - d_{p_w}(x) = 0,$$

$$x_{w,p} > 0 \Rightarrow d_{w,p}(x) - d_{p_w}(x) \ge 0$$

which is the optimality condition for the (ORP).

b) follows from the proof of part b) in Theorem 3 almost verbatim. The only difference is in the notation. Q.E.D.

Proof of Theorem 4: a) We start with the Taylor series expansion

$$J(y(t+1)) \le J(y(t)) + \alpha \sum_{i=1}^{l} \frac{\partial J}{\partial x_i}(y(t))s_i(t) + \frac{1}{2}\alpha^2 L\|s(t)\|^2 \tag{A8}$$

where L is a Lipschitz constant. For each i and t, let processor i use aggregation policy $k(i, t)$ at $x^i(t)$ at time t, then

$$\begin{aligned}
&\left\langle \frac{\partial J}{\partial x_i}(y(t)), s_i(t)\right\rangle \\
&= \left\langle \frac{\partial J}{\partial x_i}(x^i(t)), s_i(t)\right\rangle + \left\langle \frac{\partial J}{\partial x_i}(y(t)) - \frac{\partial J}{\partial x_i}(x^i(t)), s_i(t)\right\rangle \\
&= \left\langle \frac{\partial J}{\partial x_i}(x^i(t)), P_i^{k(i,t)}(x^i(t))\hat{s}_i^{k(i,t)}(t)\right\rangle \\
&\quad + \left\langle \frac{\partial J}{\partial x_i}(y(t)) - \frac{\partial J}{\partial x_i}(x^i(t)), s_i(t)\right\rangle.
\end{aligned}$$

In the second quality above, we use the fact that $s_i(t) = P_i^{k(i,t)}(x^i(t))\hat{s}_i^{k(i,t)}(t)$ where $P_i^{k(i,t)}(x^i(t))$ is the ith block submatrix of $P^{k(i,t)}(x^i(t))$ [cf. (3)]. Note that by (8) and Lemma 2b)

$$\left\langle [P_i^{k(i,t)}(x^i(t))]^T \frac{\partial J}{\partial x_i}(x^i(t)), \hat{s}_i^{k(i,t)}(t)\right\rangle \le -\|\hat{s}_i^{k(i,t)}(t)\|^2.$$

Thus,

$$\begin{aligned}
J(y(t+1)) &\le J(y(t)) - \alpha \sum_{i=1}^{l} \|\hat{s}_i^{k(i,t)}(t)\|^2 \\
&\quad + \alpha \sum_{i=1}^{l} \left\langle \frac{\partial J}{\partial x_i}(y(t)) - \frac{\partial J}{\partial x_i}(x^i(t)), s_i(t)\right\rangle \\
&\quad + \frac{L}{2}\alpha^2 \sum_{i=1}^{l} \|P_i^{k(i,t)}\hat{s}_i^{k(i,t)}(t)\|^2.
\end{aligned}$$

By Lemma 2d) we have

$$\begin{aligned}
J(y(t+1)) &\le J(y(t)) - \alpha\left(1 - \frac{L}{2}\alpha\right)\sum_{i=1}^{l} \|\hat{s}_i^{k(i,t)}(t)\|^2 \\
&\quad + \alpha \sum_{i=1}^{l} \left\langle \frac{\partial J}{\partial x_i}(y(t)) - \frac{\partial J}{\partial x_i}(x^i(t)), s_i(t)\right\rangle.
\end{aligned}$$

Let us consider the last term in the above inequality. Now

$$\left\| \frac{\partial J}{\partial x_i}(y(t)) - \frac{\partial J}{\partial x_i}(x^i(t))\right\| \le \left\| \frac{\partial J}{\partial x}(y(t)) - \frac{\partial J}{\partial x}(x^i(t))\right\|.$$

By the Lipschitz continuity of the gradient we have

$$\begin{aligned}
&\left\| \frac{\partial J}{\partial x}(y(t)) - \frac{\partial J}{\partial x}(x^i(t))\right\| \\
&\le L\|y(t) - x^i(t)\| \\
&\le L\sum_{j=1}^{N} |x_j^j(t) - x_j^i(t)| \le \alpha L \sum_{j=1}^{N} \sum_{n=t-2B}^{t-1} |s_j(n)|
\end{aligned}$$

where the second inequality comes from the inequality $(a^2 + b^2) \le (|a| + |b|)^2$, and the last inequality comes from (AC). Thus, using the fact that $|ab| \le 1/2(a^2 + b^2)$, for any scalars a, b, we have

$$\begin{aligned}
J(y(t+1)) &\le J(y(t)) - \alpha\left(1 - \frac{\alpha L}{2}\right)\sum_{i=1}^{l} \|\hat{s}_i^{k(i,t)}(t)\|^2 \\
&\quad + L\alpha^2 \sum_{i=1}^{N}\sum_{j=1}^{N}\sum_{n=t-2B}^{t-1} |s_j(n)\|s_i(t)| \\
&\le J(y(t)) - \alpha\left(1 - \frac{\alpha L}{2}\right)\sum_{i=1}^{l} \|\hat{s}_i^{k(i,t)}(t)\|^2 \\
&\quad + \frac{L\alpha^2}{2}\sum_i \sum_j \sum_n (|s_j(n)|^2 + |s_i(t)|^2) \\
&\le J(y(t)) - \alpha\left(1 - \frac{\alpha L}{2}\right)\sum_{i=1}^{l} \|\hat{s}_i^{k(i,t)}(t)\|^2 \\
&\quad + \frac{L\alpha^2 N}{2}\left[\sum_n \|s(n)\|^2 + 2B\|s(t)\|^2\right] \\
&\le J(y(t)) - \alpha\left(1 - \frac{\alpha L}{2}\right)\sum_{i=1}^{l} \|\hat{s}_i^{k(i,t)}(t)\|^2 \\
&\quad + L\alpha^2 NB\|s(t)\|^2 + \frac{L\alpha^2 N}{2}\sum_n \|s(n)\|^2.
\end{aligned}$$

Using $\|s_i(t)\|^2 = \|P_i^{i(k,t)}(x^i(t))\hat{s}_i^{k(i,t)}(t)\|^2 \le \|\hat{s}_i^{k(i,t)}(t)\|^2$ [from Lemma 2d)] and summing the above inequality over all $n = 0, 1, 2, \cdots, t$ we have

$$0 \le J(y(t+1))$$

$$\le J(y(0)) - \alpha\left(1 - \frac{\alpha L}{2}\right)\sum_{n=0}^{t}\sum_{i=1}^{l}\|\hat{s}_i^{k(i,t)}(t)\|^2$$

$$+ 2L\alpha^2 NB\sum_{n=0}^{t}\sum_{i=1}^{l}{}^{a}\|\hat{s}_i^{k(i,t)}(t)\|^2$$

$$= J(y(0)) - \alpha\left[1 - \frac{\alpha L}{2} - 2\alpha NB\right]\sum_{n=0}^{t}\sum_{i=1}^{l}{}^{a}\|\hat{s}_i^{k(i,t)}(t)\|^2.$$

For $\alpha < (L/2 + 2NB)^{-1}$, we must have $\lim_{t\to\infty}\hat{s}_i^{k(i,t)}(t) = 0$, for all i; otherwise $J(y(t)) \to -\infty$ which is impossible. Thus, $\{J(y(t))\}$ converges.

b) By (AC), we must have for all i, $\lim_{t\to\infty}\|y(t) - x^i(t)\| = 0$, i.e., the estimates at all the processors agree eventually.

c) Consider any $1 \le i \le l$. Since $\{J(y(t))\}$ converges and $J(x) \to \infty$ as $\|x\| \to \infty$, the set $\{y(t)\}$ must be bounded. Let y^* be a limit point of $\{y(t)\}$. Since the consecutive elements of each T^i differ less than B, y^* is also a limit point of the set $\{y(t)\}_{t\in T^i}$. Thus, there exists a subset $\bar{T}^i \subset T^i$ such that

$$\lim_{t\to\infty, t\in\bar{T}^i} y(t) = y^*,$$

$$\lim_{t\to\infty, t\in\bar{T}^i} s_i(t) = 0.$$

For any $1 \le k \le m$. Since the elements of T^i are bounded and each processor is using the m aggregation policies in a round-robin fashion, we can find an infinite set of times T_i^k and $T_i^k \subset \bar{T}^i$ such that T_i^k is a set of times that policy k is used. Since for any k, $P^k(\cdot)$ is continuous, and $x^k(\alpha)$ is a continuous function of x, we must have the following limits:

$$\lim_{t\to\infty, t\in T_i^k} P^k(x^i(t)) = P^k(y^*),$$

$$\lim_{t\to\infty, t\in T_i^k}\left[x_i^i(t) - \alpha\left[P^k(x^i(t))[P^k(x^i(t))]^T\frac{\partial J}{\partial x}(x(t))\right]_i\right]^+ = \left[y_i^* - \alpha\left[P^k(y^*)[P^k(y^*)]^T\frac{\partial J}{\partial x}(y^*)\right]_i\right]^+.$$

But the subsequence $\{s_i(t)\}_{t\in T_i^k}$ converging to 0; thus, we must have

$$\left[y_i^* - \alpha\left[P^k(y^*)[P^k(y^*)]^T\frac{\partial J}{\partial x}(y^*)\right]_i\right]^+ = y_i^*.$$

Since i and k are arbitrary, the above limit together with Theorem 3a) imply that y^* is a critical point of the (BP). Q.E.D.

Proof of Theorem 4a: (Sketch).

a) Following the proof of Theorem 4 we start with the Taylor series expansion

$$D(y(t+1)) \le D(y(t)) + \alpha\sum_{i=1}^{l}\frac{\partial D}{\partial x_i}(y(t))s_i(t) + \frac{1}{2}\alpha^2 L\|s(t)\|^2.$$

Continue following the proof of Theorem 4 and use k to denote $k(i, t)$. Consider the first-order term

$$\left\langle\frac{\partial D}{\partial x_i}(x^i(t)), s_i^k(t)\right\rangle$$

$$= \sum_{w\in W^i}\sum_{p\in P_w} d_p(x^i(t))s_p^k(t)$$

$$= \sum_{\hat{w}^k\in W^{k,i}}\sum_{\hat{p}^k\in\bar{P}_{\hat{w}^k}^k} d_{\hat{p}^k}(x^i(t))\, s_{\hat{p}^k}(\alpha; x^i(t))$$

$$= \sum_{\hat{w}^k\in W^{k,i}}\sum_{\hat{p}^k\in\bar{P}_{\hat{w}^k}^k, \hat{p}^k\neq\hat{p}_{\hat{w}^k}^{k,s}} d_{\hat{p}^k}(x^i(t))\, s_{\hat{p}^k}(\alpha; x^i(t))$$

$$+ \sum_{\hat{w}^k\in W^{k,i}}\left(-\sum_{\hat{p}^k\in\bar{P}_{\hat{w}^k}^k, \hat{p}^k\neq\hat{p}_{\hat{w}^k}^{k,s}} s_{\hat{p}^k}(\alpha; x^i(t))\right)$$

$$= \sum_{\hat{w}^k\in W^{k,i}}\sum_{\hat{p}^k\in\bar{P}_{\hat{w}^k}^k, \hat{p}^k\neq\hat{p}_{\hat{w}^k}^{k,s}} \bar{d}_{\hat{p}^k}(x^i(t))\, s_{\hat{p}^k}(\alpha; x^i(t))$$

$$\le -\sum_{\hat{w}^k\in W^{k,i}}\sum_{\hat{p}^k\in\bar{P}_{\hat{w}^k}^k, \hat{p}^k\neq\hat{p}_{\hat{w}^k}^{k,s}} s_{\hat{p}^k}^2(\alpha; x^i(t)) = -\|s_i^k(t)\|^2$$

where the second equality comes from (29), and last inequality comes from part b) of Theorem 3a. In the above set of equalities and inequalities, the first, second, and third equalities correspond to a change of variables to unaggregated path-flow vectors, to aggregated path-flow vectors, and to aggregated path-flow vectors in the box-constrained form, respectively.

Consider the second-order term:

$$\sum_{w\in W^i}\sum_{p\in P_w}\|s_i^k(t)\|^2 \le \sum_{\hat{w}^k\in W^{k,i}}\sum_{\hat{p}^k\in\bar{P}_{\hat{w}^k}^k} s_{\hat{p}^k}^2(\alpha; x^i(t))$$

$$= \sum_{\hat{w}^k\in W^{k,i}}\sum_{\hat{p}^k\in\bar{P}_{\hat{w}^k}^k, \hat{p}^k\neq\hat{p}_{\hat{w}^k}^{k,s}} s_{\hat{p}^k}^2(\alpha; x^i(t))$$

$$+ \sum_{\hat{w}^k\in W^{k,i}}\left(-\sum_{\hat{p}^k\in\bar{P}_{\hat{w}^k}^k, \hat{p}^k\neq\hat{p}_{\hat{w}^k}^{k,s}} s_{\hat{p}^k}(\alpha; x^i(t))\right)^2$$

$$\le A\|s_i^k(t)\|^2$$

where the first inequality comes from the fact $a^2 + b^2 \le (|a| + |b|)^2$, the second inequality comes from the fact $(a + b)^2 \le 2(a^2 + b^2)$, and $A = \max_{i,k}\max_{\hat{w}^k\in W^{k,i}}|\hat{w}^k|$. Now the rest of the proof just follows the proof of Theorem 4 almost verbatim. Q.E.D.

REFERENCES

[1] I. Y. Vakhutinsky, L. M. Dudkin, and A. A. Ryvkin, "Iterative-aggregation—A new approach to the solution of large-scale problems," *Econometrica*, vol. 47, no. 4, July 1979.

[2] W. K. Tsai and G. Huang, "Acceleration of successive approximation algorithms by residual-based and other general iterative aggregation methods," Dep. Elec. Eng., Texas A&M Univ., College Station, TX, TCSP Rep. 1988.

[3] F. Chatelin and W. L. Miranker, "Aggregation/disaggregation for eigenvalue problems," *SIAM J. Numer. Anal.*, vol. 21, June 1984.

[4] F. Chatelin and W. L. Miranker, "Acceleration by aggregation of successive approximation methods," *Linear Algebra Appl.*, vol. 43, pp. 17–42, 1982.

[5] Y. S. Arkhangel'skii, I. Y. Bakhutinskii *et al.*, "Numerical investigations of iterative aggregation methods for solving the interproduct balance problem," *Avtomatika i Telemekhanika*, pp. 75–82, July 1975.

[6] Y. A. Shpalenskii, "Iterative aggregation for unconstrained optimization problems," *Avtomatika i Telemekhanika*, pp. 97–104, Jan. 1981.

[7] C. M. Shetty and R. W. Taylor, "On large scale linear programming

by aggregation," School Indust. Syst. Eng., Georgia Inst. Technol., Atlanta, GA, Rep. J-83-4, 1983.
[8] P. J. Schweitzer, M. Puterman, and K. W. Kindle, "Iterative aggregation-disaggregation procedures for solving discounted semi-Markovian reward processes," The Graduate School of Management, Univ. Rochester, Working Paper Series 8123, 1982.
[9] I. Y. Vakhutinskii, L. M. Dudkin *et al.*, "Comparative analysis of iterative procedures for solution of the product balance problem," *Avtomatika i Telemekhanika,* pp. 124–134, Sept. 1976.
[10] L. M. Dudkin and S. A. Ivankov, "An economic and geometrical interpretation of the process of iterative aggregation," *Ekonomika i matematicheski metody,* no. 2, 1975.
[11] W. Hackbusch, *Multi-Grid Methods and Applications.* New York: Springer-Verlag, 1985.
[12] W. L. Miranker and V. Pa. Pan, "Methods of aggregation," *Linear Algebra Appl.,* vol. 29, pp. 231–257, 1980.
[13] J. N. Tsitsiklis and D. P. Bertsekas, "Distributed asynchronous optimal routing for data networks," *IEEE Trans. Automat. Contr.,* vol. AC-31, pp. 325–332, 1986.
[14] D. P. Bertsekas, "On the Goldstein–Levitin–Polyak gradient projection method," *IEEE Trans. Automat. Contr.,* vol. AC-21, pp. 174–184, Apr. 1976.
[15] D. P. Bertsekas, "Optimal routing and flow control methods for communication methods," in *Analysis and Optimization of Systems,* A. Bensoussan and J. L. Lions, Eds. New York: Springer-Verlag, pp. 615–643.
[16] J. N. Tsitsiklis, D. P. Bertsekas, and M. Athans, "Distributed asynchronous deterministic and stochastic gradient optimization algorithms," *IEEE Trans. Automat. Contr.,* vol. AC-31, pp. 803–812, Sept. 1986.
[17] M. Avriel, *Nonlinear Programming: Analysis and Methods.* Englewood Cliffs, NJ: Prentice-Hall, 1976.
[18] D. P. Bertsekas, "Distributed asynchronous computation of fixed points," *Math. Programm.,* vol. 27, pp. 107–120, 1983.
[19] W. T. Tsai, "Control and management of large and dynamic networks," Ph.D. dissertation, Dep. EECS, Univ. Calif., Berkeley, 1985.
[20] N. Shacham, "Hierarchical routing in large, dynamic ground radio," presented at the Hawaii Int. Conf. Syst. Sci., Honolulu, HI, Jan. 1985.
[21] F. Kamoun and L. Kleinrock, "Stochastic performance evaluation of hierarchical routing for large networks," *Comput. Networks,* vol. 3, pp. 337–353, 1979.
[22] W. Chu and M. Shen, "A hierarchical routing and flow control policy (HRFC) for packet radio networks," in *Computer Performance,* K. M. Chandy and M. Reiser, Eds. Amsterdam, The Netherlands: North-Holland, 1977.
[23] D. P. Bertsekas, B. Gendron, and W. K. Tsai, "Implementation of an optimal multicommodity network flow algorithm based on gradient projection and a path flow formulation," Mass. Inst. Technol., Cambridge, MA, LIDS Rep., p-1364, 1984.
[24] W. K. Tsai, "Convergence of gradient projection routing methods in an asynchronous stochastic quasi-static virtual circuit network," *IEEE Trans. Automat. Contr.,* vol. 34, pp. 20–33, Jan. 1989.
[25] G. Huang, W. K. Tsai, and W. Lu, "Fast parallel linear equation solvers based on textured decomposition," in *Proc. 26th IEEE Conf. Decision Contr.,* Los Angeles, CA, Dec. 1987.
[26] G. Huang, K. W. Lu, and J. Zaborszky, "A textured model/algorithm for computationally efficient dispatch and control on the power systems," in *Proc. 25th IEEE Conf. Decision Contr.,* Athens, Greece, 1986.
[27] J. Zarborszky, G. Huang, and K. W. Lu, "A textured model for computationally efficient reactive power control and management." *IEEE Trans. Power Apparatus Syst.,* July 1985.
[28] D. P. Bertsekas and R. G. Gallager, *Data Networks.* Englewood Cliffs, NJ: Prentice-Hall, 1987.

Wei K. Tsai (S'78–M'86), for a photograph and biography, see this issue, p. 33.

Garng Huang (S'76–M'80–SM'85) received the B.S.E.E. and M.S.E.E. degrees from the National Chiao Tung University, Hsinchu, Taiwan, R.O.C., in 1975 and 1977, respectively, and the D.Sc. degree from the Department of Systems Science and Mathematics, Washington University, St. Louis, MO, in 1980.

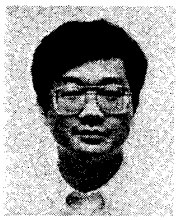
He taught at Washington University until 1984. He is currently an Associate Professor in the Department of Electrical Engineering, Texas A&M University, College Station. His research interests are in large-scale nonlinear control systems, parallel/distributed computing and control, and their applications to power systems, data networks, and distributed parameter systems. He has more than 60 publications in these areas.

Dr. Huang is a member of the Power Control Subcommittee of the IEEE PAS Society, a Technical Associate Editor designate of the IEEE TRANSACTIONS ON AUTOMATIC CONTROL, and a member of Sigma Xi. He is also a Registered Professional Engineer in Texas.

John K. Antonio (S'85–M'86–S'87–M'87) was born in Fort Worth, TX, in 1961. He received the B.S. degree (with honors) and the M.S. degree in electrical engineering from Texas A&M University, College Station, in 1984 and 1986, respectively.

He worked as a Control Systems Engineer with the Digital Flight Control Group, General Dynamics/Forth Worth Division, during the Summers of 1983 and 1984. From 1984 to 1986, he was both a Teaching Assistant and Research Assistant, while working on the Master's degree. During the Summer of 1986, he was employed by General Motors Research Laboratories, Warren, MI, as a Research Associate. Since the Fall of 1986, he has held the position of Lecturer in the Department of Electrical Engineering, Texas A&M University, where he is currently a candidate for the Ph.D. degree. His research interests include iterative A/D methods for solving large scale problems, optimal routing in data networks, parallel and distributed algorithms for solving dynamic programming problems, control theory, and power system stability and control.

Mr. Antonio is a member of the IEEE Control Systems Society and Communications Society. He is also a member of Tau Beta Pi, Eta Kappa Nu, and Phi Kappa Phi.



Wei T. Tsai (S'81–M'85) received the Ph.D. and M.S. degrees in computer science from the University of California, Berkeley, in 1982 and 1986, respectively, and the S.B. degree in computer science and engineering from the Massachusetts Institute of Technology, Cambridge, in 1979.

He is currently an Assistant Professor in the Department of Computer Science, University of Minnesota, Minneapolis. His areas of interest are software engineering, artificial intelligence, computer systems, and neural networks.