ELSEVIER

Parallel Computing 21 (1995) 1431–1450

PARALLEL COMPUTING

Distributed asynchronous algorithms with stochastic delays for constrained optimization problems with conditions of time drift

Bassem F. Beidas, George P. Papavassilopoulos *

Department of Electrical Engineering-Systems, University of Southern California, Los Angeles, CA 90089-2563, USA

Received 10 December 1992; revised 4 April 1994

Abstract

A distributed asynchronous algorithm for minimizing a function with a nonstationary minimum over a constraint set is considered. The communication delays among the processors are assumed to be stochastic with Markovian character. Conditions which guarantee the mean square and almost sure convergence to the sought solution are presented. We also present an optimal routing application for a network that connects various U.S. cities. Results of the extensive simulation that we implemented assert the practical applicability of distributed asynchronous algorithms with stochastic delays. Comparison results for varying the probability distribution of these delays are provided. The impact of varying the communication delay bound and the stepsize is also assessed.

Keywords: Optimization problem; Asynchronous iteration; Stochastic communication delay; Multiprocessor environment; Convergence analysis

1. Introduction

The recent emphasis on parallel processing is motivated by the compelling need to accelerate computations when solving large dimensional problems in which great memory storage and immense computation capabilities may hinder the performance of centralized algorithms. A number of processors are utilized that operate simultaneously in a collaborative manner on several subproblems decomposed from the original one. To further amend the enhancement of performance, the processors are permitted to communicate asynchronously such that little

* Corresponding author. Email: yorgos@nyquist.usc.edu

SSDI 0167-8191(95)00005-4

coordination of communication is maintained. It is shown that dispensing with the synchronization points at the end of each iteration induces improved efficiency, load balancing among processors and reduction of processor idle periods [4,6,9,22].

Tsitsiklis, Bertsekas and Athans [19] proposed asynchronous implementation for solving optimization problems which seems to offer the initial work of the current investigation. Since then, several algorithms suitable for operation in a multiprocessor environment emerged such that diverse areas are covered. Kushner and Yin [10] studied stochastic approximation techniques for parallel processing using the ODE approach. In their analysis, the approach of weak convergence was utilized. Kaszkurewicz, Bhaya and Šiljak [11] implemented asynchronous iterations to solve a class of nonlinear problems and derived results that retained the quasi-dominance conditions previously studied by Šiljak [16] for the synchronous case. Beidas and Papavassilopoulos [2] studied asynchronous algorithms with stochastic delays in minimizing a function with time varying minimum. Furthermore, Üresin and Dubois [23] proposed asynchronous algorithms that deal with nonnumerical methods such as symbolic computation and artificial intelligence applications.

In this paper, we study asynchronous algorithms with stochastic delays that solve minimization problems with time drifting minimum over a constraint set. The plausibility of the notion of stochastic delays stems from the fact that it models the case of an unpredictable delay in the communication among the processors and therefore addresses various reliability aspects [2]. Constrained optimization problems are prevalent in actual applications, where the nature of the problems solved necessitates imposing natural conditions. Other cases are when the designer often wishes to confine the acceptable values of processors' iterates to lie within a certain region in order to further prevent the processors from straying away from the correct solution.

Efficiency is a monumental concern regarding all algorithms. Our second main interest in this paper is to measure performance and estimate efficiency of the distributed asynchronous algorithms with stochastic delays. We also obtain comparison results of distributed asynchronous algorithms with stochastic delays and their deterministic delays counterpart algorithms of the same problem under duplicate conditions. In particular, we apply these algorithms to treat optimal routing of data communication that best exemplifies nonlinear multicommodity network flow problems. It is often the norm to find that the size of these problems is overwhelmingly large that a set of processors operating in distributed fashion is required to provide the anticipated solution.

The paper is organized as follows. In Section 2, we describe the model of asynchronous iterations for constrained minimization problems under conditions of time drift. We provide convergence conditions and analysis of the proposed models in Section 3. These conditions guarantee also convergence for the case when the problem is time invariant. In Section 4 we devise the distributed asynchronous algorithm that solves optimal routing of data communication networks. A test problem of a network that routes data among several interconnected U.S. cities and its computational results are given in Section 5. Finally, we discuss conclusions in Section 6.

2. System model

We employ a model of n processors working asynchronously on minimizing a function $F(t, x)$ subject to $x \in Q$. The minimum $x^*(t)$ is nonstationary, i.e. changes with time t. We assume that the function $F(t, x)$ is continuously differentiable and that the constraint set Q is nonempty, closed, convex and does not depend on time. Assume that the motion of the minimum is characterized by the known nonlinear function $R(t, x)$ such that

$$x^*(t+1) = R(t, x^*(t)), \tag{1}$$

and the initial value $x^*(0)$ is unknown. Knowledge of the law describing the drift was assumed by Dupac [8] and Tsypkin, Kaplinskiy and Larionov [21]. It, therefore, becomes appropriate to utilize this prediction defined in Eq. (1) in the formulation of the minimization algorithms.

We assume that the set Q is a Cartesian product of lower dimensional subsets Q_i. This entails that the projection of x on the set Q is equivalent to the projection of x_i on the set Q_i for all i which lends itself naturally to parallelization. Consequently, each processor projects independently on its constraint set. We let $Q_i \subset \mathscr{R}^{n_i}$, where $\sum_{i=1}^{n} n_i = N$ and allow each processor i to update $x_i(t)$. We denote $d_{ji}(t)$ as the delay incurred by transmitting a message from processor j to processor i at time t. We let the communication delays $\{d_{ji}(t)\}$, for all j and i, be stationary Markov chains with state space

$$S = \{1, 2, \ldots, B\},$$

where B is the maximum allowable communication delay for the transmitted messages. We let the probability transition matrix corresponding to $d_{ji}(t)$ be $P_{ji} = (p_{ji}(l, m))$, where

$$p_{ji}(l, m) = \Pr\{d_{ji}(t) = m \mid d_{ji}(t-1) = l\}, \quad \text{for } l, m = 1, 2, \ldots, B, \tag{2}$$

where here and in the sequel $\Pr\{C\}$ denotes the probability of event C.

As was done in Beidas and Papavassilopoulos [2], we utilize the vector $y^i(t)$ that summarizes the information available to each processor due to the presence of the communication delays, i.e.

$$y^i(t) = \begin{bmatrix} x_1(t+1-d_{1i}(t)) \\ \vdots \\ x_n(t+1-d_{ni}(t)) \end{bmatrix}. \tag{3}$$

Assume that instead of computing the gradient, the processors are only able to obtain a noise corrupted version of it. The asynchronous gradient projection algorithm proceeds as follows. Processor i evaluates a gradient iteration, projects back onto the set Q_i using the unique closest Euclidean distance and assumes this value as its new update.

$$\begin{aligned} \tilde{x}_i(t+1) &= R_i(t, y^i(t)) - \gamma(t)\big(\nabla_i F\big(t, R(t, y^i(t))\big) + \zeta_i(t)\big), \\ x_i(t+1) &= \Pi\big[\tilde{x}_i(t+1)\big], \end{aligned} \tag{4}$$

where

$$\Pi_i[x] = \min_{z \in Q_i} \| z - x \|. \tag{5}$$

Convergence is studied with the use of the Lyapunov function defined as the squared norm of the distance away from the desired minimum, i.e.

$$V(t, x) = \tfrac{1}{2} \| x - x^*(t) \|^2, \tag{6}$$

where here and in the sequel $\| \cdot \|$ is the Euclidean norm. Let $\mathcal{F}_t$ define the previous information of the algorithm until time t such that

$$\mathcal{F}_t = \{ d_{ji}(\tau), \zeta_i(\tau), \text{ for } \tau < t \text{ and } j, i = 1, \ldots, n \}. \tag{7}$$

$\mathcal{F}_0$ includes the initial condition information. We note that $x_i(t)$ is uniquely determined by the random variables defined by $\mathcal{F}_t$.

The basic assumptions are introduced, the form of which is expressed in terms of $y^i(t)$ which is the information available to each processor. This permits the individual verification of the basic assumptions by the various processors. It is important to note that the ability of such verification is an intrinsic property of asynchronous iterations.

Basic assumptions:

(1) There exists deterministic positive $K_1(t)$ such that for all i, we have

$$E\left[\left(y_i^i(t) - x_i^*(t) \right)' \left(\nabla_i F(t, y^i(t)) + \zeta_i(t) \right) | \mathcal{F}_t \right] \geq K_1(t) E\left[\| y^i(t) - x^*(t) \|^2 | \mathcal{F}_t \right]. \tag{8}$$

(2) There exist deterministic nonnegative $K_2(t)$ and $K_3(t)$ such that for all i, we have

$$E\left[\| \nabla_i F(t, y^i(t)) + \zeta_i(t) \|^2 | \mathcal{F}_t \right] \leq K_2(t) + K_3(t) E\left[\| y^i(t) - x^*(t+1) \|^2 | \mathcal{F}_t \right]. \tag{9}$$

(3) There exist nonnegative $\alpha(t)$ and $\beta(t)$ such that

$$(1 + \alpha(t)) \| x(t) - x^*(t) \| \leq \| R(t, x(t)) - R(t, x^*(t)) \| \leq (1 + \beta(t)) \| x(t) - x^*(t) \|. \tag{10}$$

(4) For the initial approximation, we have

$$E \| x(0) - x^*(0) \|^2 < \infty \quad \text{and} \quad \| x^*(0) \| < \infty. \tag{11}$$

Given the previous history of the algorithm, inequality (8) requires that the expected direction of $-\nabla F(t, y^i(t))$ is one of decrease with respect to the Lyapunov function $V(t, y^i(t))$. Inequality (9) imposes growth conditions on the update $\nabla_i F(t, y^i(t))$ and the noise. The inclusion of $K_2(t)$ in inequality (9) is indicative of the presence of additive noise with variance that is not necessarily finite.

3. Convergence analysis

We formulate the main convergence results of process (4). Let us denote

$$q(t) = (1+\beta(t))^2 + \gamma^2(t)K_3(t)(1+\beta(t))^2 - 2\gamma(t)K_1(t)(1+\alpha(t))^2. \tag{12}$$

For a closed and convex set Q_i, the projection operator has the following properties,

$$\begin{aligned} &(x - \Pi_i[x])'(y - \Pi_i[x]) \le 0 \quad \text{for all } y \in Q_i \\ &\| \Pi_i[x] - \Pi_i[y] \| \le \| x - y \| \quad \text{for any } x,\ y \end{aligned} \tag{13}$$

The analysis is carried out by utilizing the second property of the projection operator $\Pi_i[\cdot]$ which allows the Lyapunov function defined by Eq. (6) to maintain a supermartingale and hence the convergence is retained as in the nonconstrained case established by Beidas and Papavassilopoulos [2]. Therefore, since $x_i(t+1)$ is an orthogonal projection of $\tilde{x}_i(t+1)$ on Q_i and since $x_i^*(t+1) \in Q_i$ then

$$\begin{aligned} &\| x_i(t+1) - x_i^*(t+1) \| \\ &\quad \le \| R_i(t, y^i(t)) - R_i(t, x^*(t)) - \gamma(t)\big(\nabla_i F\big(t, R(t, y^i(t)) + \zeta_i(t)\big)\| \end{aligned} \tag{14}$$

This enables the usual Lyapunov argument to be exploited to reduce the error defined in Eq. (6) and shaping this error equation to fit the form of an easily manageable vector inequality.

We also note that a sequence $\nu(t)$ of random variables converges to a random variable ν almost surely if

$$\Pr\Big\{\lim_{t\to\infty} \nu(t) = \nu\Big\} = 1. \tag{15}$$

Theorem 1. *Consider the sequence* $\{x_i(t)\}$ *generated by Eq.* (4). *Suppose that the cost function* $F(t, x)$ *has a unique minimum at* $x = x^*(t) \in Q$ *for any* t. *Let the basic assumptions* (1)–(4) *be satisfied. In addition, assume that*

(1) $\sum_{t=0}^{\infty} q(t) < \infty, \quad q(t) \ge 0,$

(2) $\sum_{t=0}^{\infty} \beta(t) < \infty,$

(3) $\sum_{t=0}^{\infty} \| R(t, 0) \|^2 < \infty,$

(4) $\sum_{t=0}^{\infty} \gamma^2(t) K_2(t) < \infty.$

Then for every initial condition the sequence $\{x_i(t)\}$ *converges to* $x_i^*(t) \in Q_i$ *in the mean square and almost surely for each* i.

Proof. Subtracting $x_i^*(t+1)$ from Eq. (4) and taking norms, we write

$$\begin{aligned}&\| x_i(t+1) - x_i^*(t+1) \|^2 \\ &\quad = \| \Pi_i\big[R_i(t, y^i(t)) - \gamma(t)\big(\nabla_i F\big(t, R(t, y^i(t)) + \zeta_i(t)\big)\big] \\ &\qquad - R_i(t, x^*(t)) \|^2. \end{aligned} \tag{16}$$

Using the properties of the projection operator and recalling the fact that $x_i^*(t) \in Q_i$, we write

$$\begin{aligned}&\| x_i(t+1) - x_i^*(t+1) \|^2 \\ &\quad \le \| R_i(t, y^i(t)) - R_i(t, x^*(t)) \|^2 + \gamma^2(t) \| \nabla_i F\big(t, r(t, y^i(t)) + \zeta_i(t) \|^2 \\ &\qquad - 2\gamma(t)\big(R_i(t, y^i(t)) - R_i(t, x^*(t))\big)'\big(\nabla, F\big(t, R(t, y^i(t)) + \zeta_i(t)\big).\end{aligned} \tag{17}$$

Here, we notice that applying steps similar to those of the proofs contained in Beidas and Papavassilopoulos [2] yields the required result. □

It is worthy of mention that convergence in the mean square requires a weaker version of condition 4, which can be replaced by $\lim_{t\to\infty} \gamma^2(t) K_2(t) = 0$.

Next we cover different cases of the projection operator as the constraint set Q_i takes more specific forms.

Example 1. Let the set Q_i consist of simple constraints such that

$$Q_i = \{ x_i : x_i \ge 0 \}.$$

In this case, the asynchronous gradient projection algorithm is described as

$$x_i(t+1) = \max\big\{0, R_i(t, y^i(t)) - \gamma(t)\big(\nabla_i F\big(t, R(t, y^i(t))\big) + \zeta_i(t)\big)\big\}. \tag{18}$$

Example 2. Let the set Q_i consist of upper and lower bounds such that

$$Q_i = \{ x_i : a_i \le x_i \le b_i \},$$

where a_i and $b_i \in \mathscr{R}^{n_i}$. In this case, the asynchronous gradient projection algorithm takes the form of

$$x_i(t+1) = \begin{cases} a_i & \tilde{x}_i(t+1) < a_i \\ b_i & \tilde{x}_i(t+1) > b_i \\ R_i(t, y^i(t)) - \gamma(t) & \\ \quad \times \big(\nabla_i F\big(t, R(t, y^i(t))\big) + \zeta_i(t)\big) & \text{otherwise.} \end{cases} \tag{19}$$

4. Routing network

Suppose that we are given a network of nodes and arcs and a set W of ordered pairs w of distinct nodes referred to as the origin-destination (OD) of w. We are also given that r_w (measured in data units/seconds) is the arrival rate of traffic entering the network at the origin and exiting at the destination of w. We denote P_w as the set of all simple paths that connect the origin and destination of w, x_p as the flow routed through path p and x_w as all the path flows $x_p \in P_w$. The fundamental constraints imposed on this problem require the conservation of the load when shared among the various paths and maintaining the nonnegativity property of the path flows which yield

$$\sum_{p \in P_w} x_p = r_w, \quad \forall w \in W, \tag{20}$$

$$x_p \geq 0, \quad \forall p \in P_w, \quad \forall w \in W. \tag{21}$$

Define F_{ij} as the total flow of arc (i, j)

$$F_{ij} = \sum_{\text{all paths containing } (i,j)} x_p. \tag{22}$$

Consider a cost function such that

$$D(x) = \sum_{(i,j)} D_{ij}(F_{ij}). \tag{23}$$

Our objective is to find the set of paths for each origin-destination pair and the amount of flow routed along each path such that the cost function defined in Eq. (23) is minimized.

The distributed asynchronous algorithm of this section is carried out along the lines of the one studied by Tsitsiklis and Bertsekas [18]. It takes the gradient projection form in which processor w is responsible for updating x_w. At every step the update is in the direction opposite to the gradient of the cost function and whenever a processor detects that its iteration is excluded from the constraint set, it enforces feasibility by projecting its iteration back onto the feasible set.

In a manner reminiscent of the ARPANET algorithm [13], the end node of any arc ascertains the amount of flow through that arc by averaging of some previous values of the total flow of the arcs F_{ij}. We therefore write

$$\tilde{F}_{ij}(t) = \sum_{t'=t-T}^{t} c_{ij}(t, t') F_{ij}(t'), \tag{24}$$

where $c_{ij}(t, t')$ arc nonnegative scalars such that

$$\sum_{t'=t-T}^{t} c_{ij}(t, t') = 1$$

and T is the bound over which the averaging is implemented. The averaged value of the total flow of arcs $\tilde{F}_{ij}$ is then propagated to all other nodes. Due to the

distributed asynchronous nature of the algorithm, this information may be received with a delay. Consequently, processor w computes

$$\lambda_p(t) = \sum_{(i,j)\in p} D'_{ij}\left(\tilde{F}_{ij}(t+1-d^w(t)\right), \tag{25}$$

where $'$ denotes the derivative and λ_w is all the λ_p for the paths $p \in P_w$. We note that $d^w(t)$ is the communication delay encountered when sending information to processor w at time t. The next step casts the equality constraints in a form that is forthcoming to parallelization by means of transforming the equality constraints to nonnegativity constraints. Then each processor would be able to project onto the positive orthant at its own pace and independently of the other processors. Consequently, processor w finds the minimum first derivative length (MFDL) [5] path $\tilde{p}_w(t)$ such that

$$\tilde{\lambda}_{\tilde{p}_w(t)}(t) = \min_{p\in P_w} \lambda_p(t) \tag{26}$$

In practice, there exist transients in the total flows F^{ij} that occur as the routing changes. This contributes to having a settling time quite substantial to be ignored that renders processor w incapable of computing actual flows. Processor w will in turn use $\lambda_w(t)$ to compute desired path flows $\bar{x}_w(t)$, whose components are $\bar{x}_p(t)$ for $p \in P_w$. Therefore, the actual values do not assume their desired values instantaneously. Instead, the actual flow $x_p(t)$ takes some value between $\bar{x}_p(t)$ and $x_p(t-1)$. We assume that there exist scalars $\alpha > 0$, $a_p(t)$, such that

$$a_p(t) \geq \alpha, \quad \forall p, t, \tag{27}$$

and

$$x_p(t+1) = a_p(t)\bar{x}_p(t) + \left(1 - a_p(t)\right)x_p(t), \quad \forall p, t. \tag{28}$$

The distributed asynchronous algorithm is described as follows. For any $p \in P_w$, $p \neq \tilde{p}_w(t)$

$$\bar{x}_p(t) = \max\left\{0,\ x_p(t) - \frac{\gamma}{H_p(t)}\left(\lambda_p(t) - \tilde{\lambda}_{\tilde{p}_w(t)}(t)\right)\right\}, \tag{29}$$

where H_p is the second derivative length

$$H_p(t) = \sum_{(i,j)\in L_p} D''_{ij}\left(F_{ij}(t)\right) \tag{30}$$

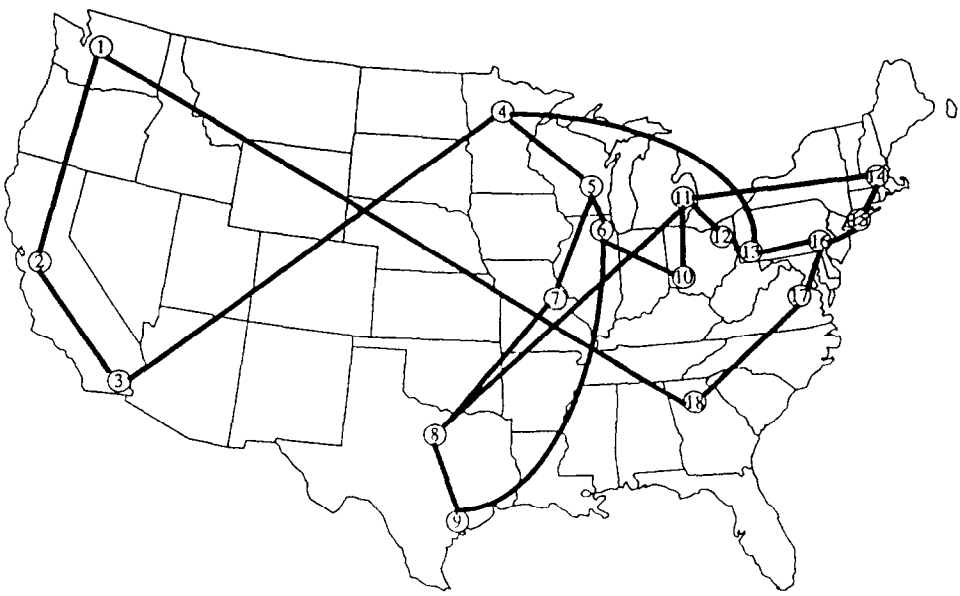
and L_p is the set of arcs belonging to either p or the corresponding minimum first derivative path $\tilde{p}_w$, but not both. For $p = \tilde{p}_w(t)$, we have

$$\bar{x}_{\tilde{p}_w(t)}(t) = r_w - \sum_{p\in P_w, p\neq\tilde{p}_w(t)} \bar{x}_p(t). \tag{31}$$

5. Simulation studies

In this section we use computer simulation to measure performance and estimate efficiency of distributed asynchronous algorithms with stochastic delays in the context of data communication optimal routing networks.

In our test problem, we considered the network topology depicted in Fig. 1 where all the connections are assumed to carry the flows bidirectionally. This network connects various U.S. cities with the intention of routing data as follows.
r_1: load to be routed from Seattle to Detroit
r_2: load to be routed from Detroit to Seattle



1. Seattle
2. San Francisco
3. Los Angeles
4. Minneapolis
5. Milwaukee
6. Chicago
7. St. Louis
8. Dallas
9. Houston
10. Cincinatti
11. Detroit
12. Cleveland
13. Pittsburg
14. Boston
15. New York
16. Philadelphia
17. Washington
18. Atlanta

Fig. 1. The network topology.

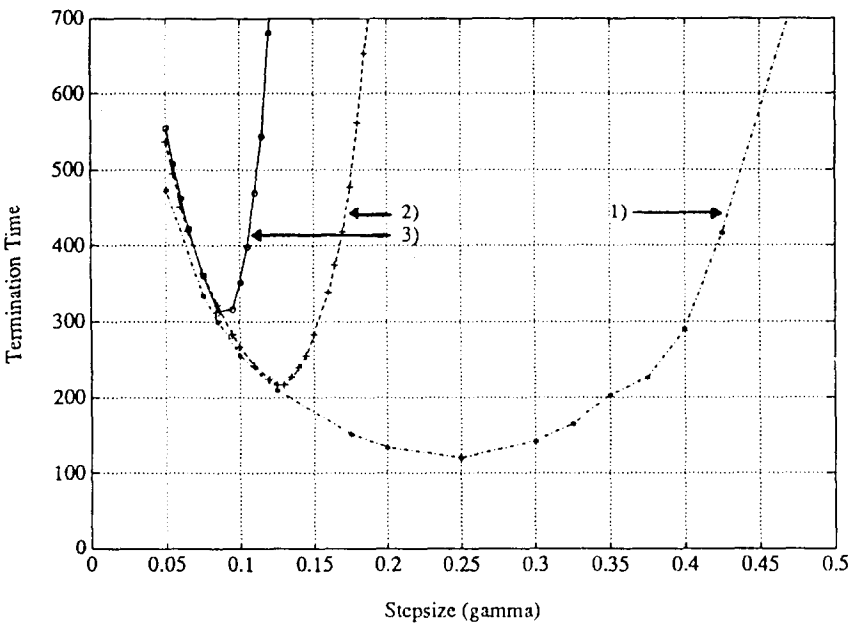
r_3: load to be routed from Chicago to Washington
r_4: load to be routed from Washington to Chicago
r_5: load to be routed from Houston to Atlanta
r_6: load to be routed from Atlanta to Houston.

In all the simulations, we considered the cost function at every arc to be quadratic

$$D_{ij} = \tfrac{1}{2}(F_{ij})^2, \quad \text{for every } (i, j), \tag{32}$$

the traffic loads r_w for all w were equal to 3.0 and the parameter T in Eq. (24) was chosen to be 4. In addition, the parameter α in Eq. (27) was chosen to be 0.25. The distributed asynchronous algorithms with stochastic delays were tested on different initial conditions to show the uniqueness of the minimum point.

In our simulation, the delays $d^w(t)$ assume their values from the set $\{1, 2, \cdots, B\}$, where B is the communication delay bound and the delays were either sequences



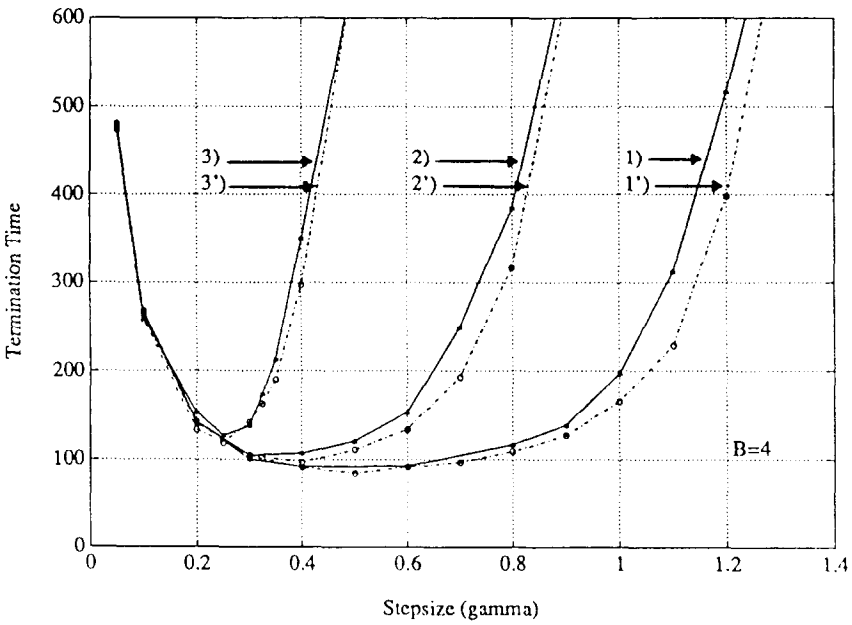
1) Distributed asynchronous algorithms with $B = 4$
2) Distributed asynchronous algorithms with $B = 8$
3) Distributed asynchronous algorithms with $B = 12$

Fig. 2. Performance curves for distributed asynchronous algorithms with bounded delays for different communication delay bounds. 100 runs were considered.

of Markov chains or independently generated with probabilities equal to the limiting behavior of these Markov delays. The delays were generated according to several probability distributions and the initial delays were chosen at random with equal probabilities. The probability distributions of these delays are relegated in the Appendix 2.

It is assumed that the algorithm terminates at time t if the termination function $\mathrm{TF}(t)$ meets the tolerance level of 0.001, i.e.

$$\mathrm{TF}(t) = \max_{\tau,\tau' \in \{t-B+1,\cdots,t\}} E \| x(\tau) - x(\tau') \| \leq 0.001 \tag{33}$$

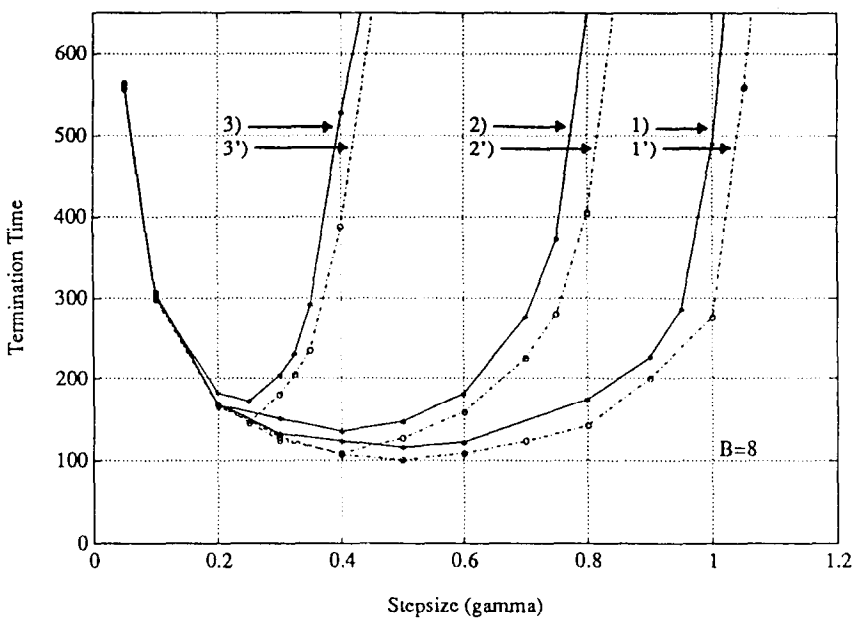


1) Algorithms with Markov delays for probability distribution $PD1$
1') Algorithms with independent delays whose probabilities equal the limiting behavior of $PD1$
2) Algorithms with Markov delays for probability distribution $PD2$
2') Algorithms with independent delays whose probabilities equal the limiting behavior of $PD2$
3) Algorithms with Markov delays for probability distribution $PD3$
3') Algorithms with independent delays whose probabilities equal the limiting behavior of $PD3$

Fig. 3. Performance curves for distributed asynchronous algorithms with stochastic delays under different probability distributions. $B = 4$ and 100 runs were considered.

We computed $E\|\cdot\|$ by averaging over 100 trials of the experiment. For each trial a different computer realization of the delays was used. Since the previous values of the iterations that lie within the delay bound affect the value at which the algorithm stands, the above termination criterion is necessary to ensure that all of these values have also been stabilized.

We started by simulating the behavior of distributed asynchronous algorithms with bounded delays. In essence, the delays were independently generated from a uniform distribution on the set $\{1, 2, \cdots, B\}$. Fig. 2 plots the performance of these algorithms as measured by the termination time in terms of varying the stepsize for the delay bounds $B = 4$, 8 and 12.



1) Algorithms with Markov delays for probability distribution $PD1$
1′) Algorithms with independent delays whose probabilities equal the limiting behavior of $PD1$
2) Algorithms with Markov delays for probability distribution $PD2$
2′) Algorithms with independent delays whose probabilities equal the limiting behavior of $PD2$
3) Algorithms with Markov delays for probability distribution $PD3$
3′) Algorithms with independent delays whose probabilities equal the limiting behavior of $PD3$

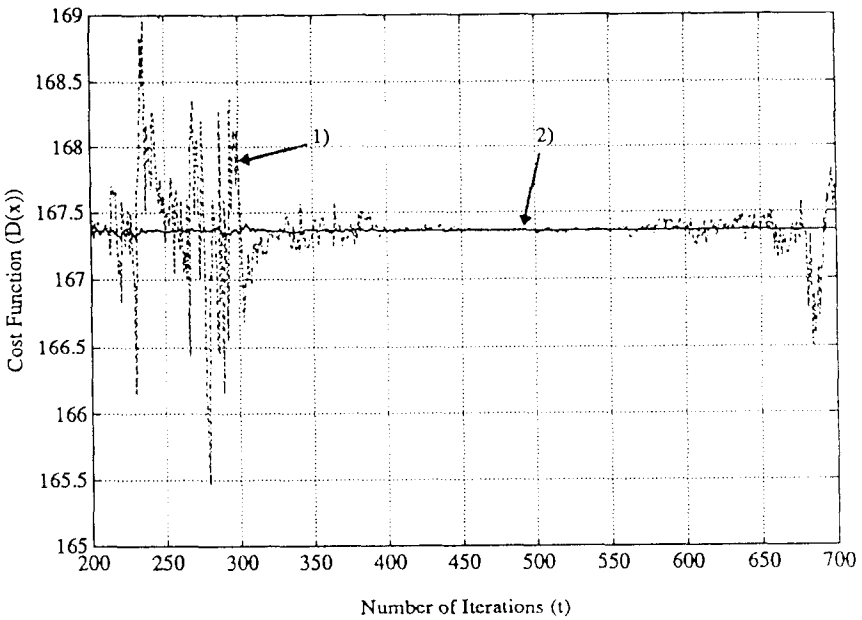
Fig. 4. Performance curves for distributed asynchronous algorithms with stochastic delays under different probability distributions. $B = 8$ and 100 runs were considered.

Next we tested the case of ordered scheduling that is discussed in Beidas and Papavassilopoulos [3]. A reasonable restriction is imposed where we assume that the information is received in the order it was produced. We assume that each processor w has a local memory where the latest x_w generated at time instant t is kept and when the new information arrives it is labelled using a time stamp as to when it was computed by the other processors. If it happens that this processor acknowledges that the information it receives was generated at a time instant earlier than t, then processor w will discard it. Therefore, the probability distribution enforces that

$$\Pr\{d^w(t) > d^w(t-1) + 1\} = 0, \quad \text{for all } w \text{ and } t, \tag{34}$$

which entails that the entries of the probability matrices for the delays that are above the superdiagonal to be zeros.

We simulated the distributed asynchronous algorithms with Markov delays having the property defined by equation (34) and the distributed asynchronous



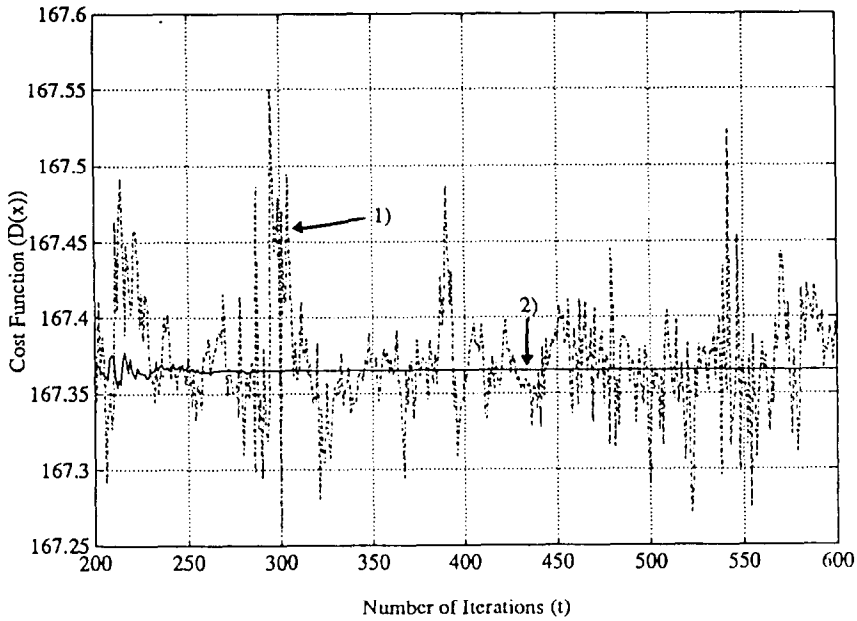
1) Algorithms with Markov delays
2) Algorithms with independent delays

Fig. 5. Performance curves for distributed asynchronous algorithms with stochastic delays under *PD*1 for $\gamma = 1.095$. $B = 8$ and 1 run were considered.

algorithms with independent delays whose probabilities are equal to the limiting behavior of these Markov delays. Fig. 3 and Fig. 4 illustrate the performance of these algorithms for different probability distributions and for $B = 4$ and $B = 8$, respectively.

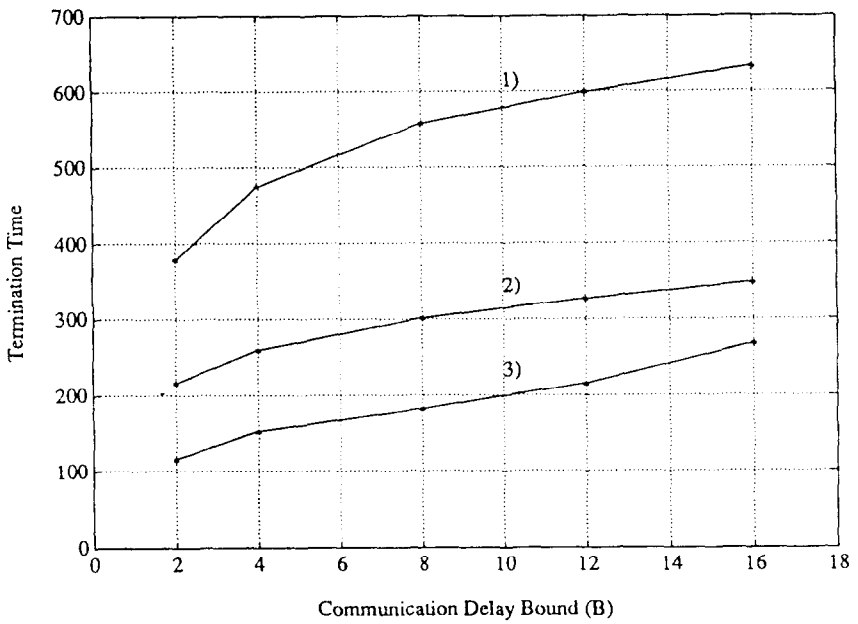
In general, we make the following observations. Firstly, for small stepsize γ, changing the probability distribution has little discernible effect on the termination time. Secondly, the performance curves of the algorithms with Markov delays follow those of the algorithms with independent delays, corroborating our intuition. Thirdly, the performance curve which corresponds to the probability distribution *PD*1 that places emphasis on recent values of the delays spans more values of stepsizes before the eventual divergence.

From Fig. 3 and Fig. 4, we notice that the performance curves of the Markov delays lag those of the independent delays. This is attributed to the fact that the time needed to bring the Markov delays to their limiting behavior causes the algorithms with Markov delays to converge more slowly than the algorithms with



1) Algorithms with Markov delays.
2) Algorithms with independent delays.

Fig. 6. Performance curves for distributed asynchronous algorithms with stochastic delays under *PD*2 for $\gamma = 0.845$. $B = 8$ and 1 run were considered.



1) Algorithms with Markov delays for stepsize $\gamma = 0.05$
2) Algorithms with Markov delays for stepsize $\gamma = 0.1$
3) Algorithms with Markov delays for stepsize $\gamma = 0.2$

Fig. 7. Effects of stepsize and delay bound on performance for distributed asynchronous algorithms with Markov delays. 100 runs were considered.

independent delays. This effect is more prevailing for the case of $B = 8$ where for some stepsizes the algorithms with independent delays converge while their Markov delay counterparts do not.

To underscore the above phenomenon, we obtained Fig. 5 and Fig. 6 by plotting one run of the performance of algorithms with Markov delays for $\gamma = 1.095$, probability distribution $PD1$ and $\gamma = 0.845$, probability distribution $PD2$. The same was done for the algorithms with independent delays. From both figures we notice that while the independent delays case converges to the minimum value of the cost function, the Markov delays case exhibits severe oscillatory behavior around the minimum.

Next, we examined the effects of the stepsize γ and the communication delay bound B on distributed asynchronous algorithms with Markov delays. Fig. 7 depicts the termination time as the delay bound B and stepsize γ are varied. Noteworthy of mention is that Fig. 3 and Fig. 4 show that algorithms with stepsize

γ chosen to be 0.05, 0.1 and 0.2 display constant termination time as the probability distribution is changed. We note that the termination time grows quickly with decreasing stepsize γ. In addition, it grows faster with increasing delay bound B when stepsize γ is small than it does when γ is large.

In summary,

- Fig. 1 shows the performance of asynchronous algorithm when using deterministic delays for different delay bounds. Fig. 2 shows the performance of the asynchronous algorithm when using stochastic delays with delay bound $B = 4$. A comparison between these figures shows that the algorithm with stochastic delays sustains convergence for larger values of the stepsize γ (i.e., $\gamma \in (0, 1.3)$) where the algorithm for deterministic delays converges only for $\gamma \in (0, 0.5)$.
  This observation is even more prevalent when the delay bound is large. A comparison of graph 2) of Fig. 1 with that of Fig. 3 for $B = 8$ shows that algorithm with deterministic delays converges only when $\gamma \in (0, 0.2)$ while the algorithm with stochastic delays with a certain probability distribution converges for $\gamma \in (0, 1.1)$
  This leads us to conclude that the region of convergence is larger when using stochastic delays.
- Now that we have shown the merits gained when using stochastic delays, we included Fig. 5–Fig. 7 to discuss performance issues of using certain type of stochastic delays over another. In particular, we show the performance of the algorithm with independent delays versus Markovian delays.
- Fig. 7 displays the performance of the asynchronous algorithm when other variables are changed. In particular, the effect of stepsize γ and the communication delay bound B on the performance of the asynchronous algorithm with Markovian Delays.

6. Conclusion

We studied the behavior of distributed asynchronous iterations with stochastic delays that solve optimization problems with nonstationary minimum over a constrained set. For the purpose of confining the iterates to the constraint set, each processor evaluates a gradient iteration and then projects back its iterate independently of the other processors. This procedure guarantees that each iterate generated by the algorithm is contained in the constraint set. The analysis that establishes the sufficiency conditions required to guarantee mean square and almost sure convergence is based upon utilizing a Lyapunov function given by Eq. (6) and using properties of the projection that maintain its supermartingale property and, finally, showing that the adverse effects possibly inflicted by the communication delays are negligible.

With the aid of simulation studies we estimated the performance of distributed asynchronous iterations with stochastic delays and obtained comparison results as the probability distribution of these delays changed. We also assessed the impact

of varying the communication delay bound and the stepsize on the termination of these algorithms and it was shown that the performance of distributed asynchronous algorithms is in fact predictable.

Appendix 1

- Almost sure convergence (with probability 1) is a form of convergence of random variables. A sequence $X(t)$ of random variables is said to converge to a random variable X, almost surely (with probability 1), if

$$\Pr\left(\lim_{t\to\infty} X(t) = X\right) = 1,$$

  where $\Pr(A)$ indicates the probability of event A.
- A sequence of random variables $X(t)$ is said to be supermartingale if

$$E(X(t+1) \mid X(0),\ldots,X(t)) \le X(t), \quad EX(0) < \infty,$$

  where $E(X(t+1)|X(0),\ldots,X(t))$ is the conditional mathematical expectation of $X(t+1)$ for the given $X(0),\ldots,X(t)$. A supermartingale is a generalization to the stochastic case of the notion of monotonically decreasing sequence.
- The definition of positive orthant is the set

$$x \in \mathscr{R}^n : x \ge 0.$$

Appendix 2

We provide the probability distribution for the delays that result from the distributed asynchronous nature of the algorithms when solving the optimal routing network of U.S. cities given in Section 5.

First, when the communication delay bound $B = 4$, the probability matrices that characterize probability distribution $PD1$ are given below.

$$P^1 = P^4 = \begin{bmatrix} 0.75 & 0.25 & 0 & 0 \\ 0.65 & 0.2 & 0.15 & 0 \\ 0.6 & 0.2 & 0.1 & 0.1 \\ 0.25 & 0.25 & 0.25 & 0.25 \end{bmatrix}, \quad P^2 = P^5 = \begin{bmatrix} 0.6 & 0.4 & 0 & 0 \\ 0.5 & 0.2 & 0.3 & 0 \\ 0.25 & 0.25 & 0.25 & 0.25 \\ 0.625 & 0.125 & 0.125 & 0.125 \end{bmatrix}$$

$$P^3 = P^6 = \begin{bmatrix} 0.8 & 0.2 & 0 & 0 \\ 0.4 & 0.3 & 0.3 & 0 \\ 0.5 & 0.2 & 0.2 & 0.1 \\ 0.1 & 0.34 & 0.25 & 0.31 \end{bmatrix}$$

Second, the probability matrices that characterize probability distribution $PD2$ are given below.

$$(P^i,\ i=1,\ldots,6)=\begin{bmatrix}0.5 & 0.5 & 0 & 0\\ 0.4 & 0.3 & 0.3 & 0\\ 0.25 & 0.25 & 0.25 & 0.25\\ 0.25 & 0.25 & 0.25 & 0.25\end{bmatrix}$$

Third, the probability matrices that characterize probability distribution $PD3$ are given below.

$$P^1=P^4=\begin{bmatrix}0.2 & 0.8 & 0 & 0\\ 0.2 & 0.25 & 0.55 & 0\\ 0.1 & 0.2 & 0.1 & 0.6\\ 0.25 & 0.25 & 0.25 & 0.25\end{bmatrix},\quad P^2=P^5=\begin{bmatrix}0.375 & 0.625 & 0 & 0\\ 0.3 & 0.2 & 0.5 & 0\\ 0.25 & 0.25 & 0.25 & 0.25\\ 0.125 & 0.125 & 0.125 & 0.625\end{bmatrix}$$

$$P^3=P^6=\begin{bmatrix}0.5 & 0.5 & 0 & 0\\ 0.1 & 0.3 & 0.6 & 0\\ 0.1 & 0.2 & 0.2 & 0.5\\ 0.1 & 0.34 & 0.25 & 0.31\end{bmatrix}$$

Now we provide the probability distributions when the communication delay bound $B=8$. First, the probability matrices that characterize probability distribution $PD1$ are given below.

$$P^1=P^4=\begin{bmatrix}0.75 & 0.25 & 0 & 0 & 0 & 0 & 0 & 0\\ 0.55 & 0.1 & 0.35 & 0 & 0 & 0 & 0 & 0\\ 0.25 & 0.25 & 0.25 & 0.25 & 0 & 0 & 0 & 0\\ 0.33 & 0.33 & 0.21 & 0.06 & 0.07 & 0 & 0 & 0\\ 0.1 & 0.15 & 0.25 & 0.25 & 0.2 & 0.05 & 0 & 0\\ 0.3 & 0.15 & 0.15 & 0.1 & 0.1 & 0.1 & 0.1 & 0\\ 0.125 & 0.125 & 0.125 & 0.125 & 0.125 & 0.125 & 0.125 & 0.125\\ 0.4 & 0 & 0 & 0.2 & 0.1 & 0.1 & 0.1 & 0.1\end{bmatrix}$$

$$P^2=P^5=\begin{bmatrix}0.5 & 0.5 & 0 & 0 & 0 & 0 & 0 & 0\\ 0.45 & 0.3 & 0.25 & 0 & 0 & 0 & 0 & 0\\ 0.25 & 0.25 & 0.25 & 0.25 & 0 & 0 & 0 & 0\\ 0.625 & 0.125 & 0.125 & 0.125 & 0 & 0 & 0 & 0\\ 0.3 & 0.3 & 0.2 & 0.05 & 0.05 & 0.1 & 0 & 0\\ 0.15 & 0.15 & 0.15 & 0.15 & 0.15 & 0.15 & 0.1 & 0\\ 0.125 & 0.125 & 0.125 & 0.125 & 0.125 & 0.125 & 0.125 & 0.125\\ 0.3 & 0.1 & 0.1 & 0.1 & 0.1 & 0.1 & 0.1 & 0.1\end{bmatrix}$$

$$P^3=P^6=\begin{bmatrix}0.8 & 0.2 & 0 & 0 & 0 & 0 & 0 & 0\\ 0.4 & 0.3 & 0.3 & 0 & 0 & 0 & 0 & 0\\ 0.2 & 0.25 & 0.25 & 0.3 & 0 & 0 & 0 & 0\\ 0.1 & 0.24 & 0.25 & 0.31 & 0.1 & 0 & 0 & 0\\ 0.2 & 0.2 & 0.2 & 0.1 & 0.1 & 0.2 & 0 & 0\\ 0.2 & 0.3 & 0.1 & 0.05 & 0.05 & 0.15 & 0.15 & 0\\ 0.2 & 0.2 & 0.1 & 0.1 & 0.1 & 0.1 & 0.1 & 0.1\\ 0.125 & 0.125 & 0.125 & 0.125 & 0.125 & 0.125 & 0.125 & 0.125\end{bmatrix}$$

Second, the probability matrices that characterize the probability distribution $PD2$ are given below.

$$P^i = \begin{bmatrix} 0.5 & 0.5 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0.4 & 0.3 & 0.3 & 0 & 0 & 0 & 0 & 0 \\ 0.25 & 0.25 & 0.25 & 0.25 & 0 & 0 & 0 & 0 \\ 0.2 & 0.2 & 0.2 & 0.2 & 0.2 & 0 & 0 & 0 \\ 0.1667 & 0.1667 & 0.1667 & 0.1667 & 0.1667 & 0.1665 & 0 & 0 \\ 0.1429 & 0.1429 & 0.1429 & 0.1429 & 0.1429 & 0.1429 & 0.1426 & 0 \\ 0.125 & 0.125 & 0.125 & 0.125 & 0.125 & 0.125 & 0.125 & 0.125 \\ 0.125 & 0.125 & 0.125 & 0.125 & 0.125 & 0.125 & 0.125 & 0.125 \end{bmatrix},$$

for $i = 1, \ldots, 6$.

Third, the probability matrices that characterize probability distribution $PD3$ are given below.

$$P^1 = P^4 = \begin{bmatrix} 0.2 & 0.8 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0.35 & 0.1 & 0.55 & 0 & 0 & 0 & 0 & 0 \\ 0.25 & 0.25 & 0.25 & 0.25 & 0 & 0 & 0 & 0 \\ 0.03 & 0.33 & 0.21 & 0.06 & 0.37 & 0 & 0 & 0 \\ 0.1 & 0.15 & 0.05 & 0.25 & 0.2 & 0.25 & 0 & 0 \\ 0.1 & 0.15 & 0.15 & 0.1 & 0.1 & 0.2 & 0.2 & 0 \\ 0.125 & 0.025 & 0.025 & 0.025 & 0.025 & 0.025 & 0.125 & 0.625 \\ 0.2 & 0 & 0 & 0.1 & 0.1 & 0.1 & 0.1 & 0.4 \end{bmatrix}$$

$$P^2 = P^5 = \begin{bmatrix} 0.3 & 0.7 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0.25 & 0.1 & 0.65 & 0 & 0 & 0 & 0 & 0 \\ 0.15 & 0.05 & 0.35 & 0.45 & 0 & 0 & 0 & 0 \\ 0.225 & 0.125 & 0.125 & 0.525 & 0 & 0 & 0 & 0 \\ 0.1 & 0 & 0.2 & 0.05 & 0.05 & 0.6 & 0 & 0 \\ 0.1 & 0.1 & 0 & 0 & 0.15 & 0.15 & 0.5 & 0 \\ 0.125 & 0.025 & 0.025 & 0.025 & 0.025 & 0.025 & 0.125 & 0.625 \\ 0.05 & 0.05 & 0.05 & 0.05 & 0.15 & 0.05 & 0.05 & 0.55 \end{bmatrix}$$

$$P^3 = P^6 = \begin{bmatrix} 0.4 & 0.6 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0.2 & 0.1 & 0.7 & 0 & 0 & 0 & 0 & 0 \\ 0.2 & 0.25 & 0.1 & 0.45 & 0 & 0 & 0 & 0 \\ 0.05 & 0.24 & 0 & 0.31 & 0.4 & 0 & 0 & 0 \\ 0.2 & 0.2 & 0.2 & 0.1 & 0.1 & 0.2 & 0 & 0 \\ 0.2 & 0.15 & 0.1 & 0.05 & 0.05 & 0.15 & 0.3 & 0 \\ 0.05 & 0.2 & 0.1 & 0.1 & 0.1 & 0.1 & 0.1 & 0.25 \\ 0.05 & 0.05 & 0.05 & 0.05 & 0.05 & 0.05 & 0.075 & .625 \end{bmatrix}.$$

References

[1] G.M. Baudet, Asynchronous iterative methods for multiprocessors, *J. ACM* 25, (1978) 226–244.

[2] B.F. Beidas and G.P. Papavassilopoulos, Asynchronous implementation with stochastic delays of optimization algorithms under conditions of drift, submitted to *IEEE Trans. Auto. Cont.*

[3] B.F. Beidas and G.P. Papavassilopoulos, Convergence analysis of linear asynchronous iterations with stochastic delays, *Parallel Comput.* 19 (1993) 281–302.
[4] D.P. Bertsekas, Distributed dynamic programming, *IEEE Trans. Auto. Cont.* AC-27 (1982) 610–616.
[5] D.P. Bertsekas and R. Gallager, *Data Networks* (Prentice Hall, New Jersey, 1987).
[6] D.P. Bertsekas and J.N. Tsitsiklis, *Parallel and Distributed Computation, Numerical Methods* (Prentice Hall, New Jersey, 1989).
[7] D. Chazan and W. Miranaker, Chaotic relaxation, *Linear Algebra Appl.* 2 (1969) 199–222.
[8] V.A. Dupac, A dynamic stochastic approximation method, *Ann. Math. Statistics* 36 (1965) 1695–1702.
[9] H.T. Kung, Synchronized and asynchronous parallel algorithms for multiprocessors, in: J.F. Traub, ed., *Algorithms and Complexity: New Directions and Recent Results* (Academic Press, 1976) 153–200.
[10] H.J. Kushner and G. Yin, Asymptotic properties of distributed and communicating stochastic approximation algorithms, *SIAM J. Cont. Optimiz.* 25 (1987) 1266–1290.
[11] E. Kaszkurewicz, A. Bhaya and D.D. Šiljak, On the convergence of parallel asynchronous block-iterative computations, *Linear Algebra Appl.* 131 (1990) 139–160.
[12] D.G. Luenberger, *Linear and Nonlinear Programming* (Addison-Wesley, 1984).
[13] J.M. McQuillan, I. Richer and E.C. Rosen, The new routing algorithm for the ARPANET, *IEEE Trans. Commun.* COM-28 (1980) 711–719.
[14] B.T. Polyak, Convergence and convergence rate of iterative algorithms, Part I: General case, *Automation Remote Control* 37 (1976) 83–94.
[15] B.T. Polyak and Ya.Z. Tsypkin, Pseudogradient adaptation and training algorithms, *Automation Remote Control* 34 (1973) 45–68.
[16] D.D. Šiljak, *Large-Scale Dynamic systems* (North-Holland, New York, 1978).
[17] P. Tseng, D.P. Bertsekas and J.N. Tsitsiklis, Partially asynchronous, parallel algorithms for network flow and other problems, *SIAM J. Cont. Optimiz.* 28 (1990) 678–710.
[18] J.N. Tsitsikils and D.P. Bertsekas, Distributed asynchronous optimal routing in data networks, *IEEE Trans. Auto. Cont.* AC-31 (1986) 325–332.
[19] J.N. Tsitsiklis, D.M. Bertsekas and M. Athans, Distributed asynchronous deterministic and stochastic gradient optimization algorithms, *IEEE Trans. Auto. Cont.* AC-31 (1986) 803–812.
[20] J.N. Tsitsiklis and G.D. Stamoulis, On the average communication complexity of asynchronous distributed algorithms, Report LIDS-P-1986, Laboratory for Information and Decision Systems, M.I.T., 1990.
[21] Ya.Z. Tsypkin, A.I. Kaplinskiy and K.A. Larionov, Adaptation and learning algorithms under nonstationary conditions, *Engineering Cybernet.* 5 (1970) 829–840.
[22] A. Üresin, Asynchronous iterative algorithms for problems with discrete data, Ph.D. Thesis, University of Southern California, Los Angeles, CA, 1990.
[23] A. Üresin and M. Dubois, Parallel asynchronous algorithms for discrete data, *J. ACM* 37 (1990) 588–606.