OCTOBER, 1984

LIDS-P-1412
(Revision of LIDS-P-1342)

CONVERGENCE THEORIES OF DISTRIBUTED ITERATIVE PROCESSES: A SURVEY[†]

by

Dimitri P. Bertsekas*
John N. Tsitsiklis*
Michael Athans*

ABSTRACT

We consider a model of distributed iterative algorithms whereby several processors participate in the computation while collecting, possibly stochastic information from the environment or other processors via communication links. Several applications in distributed optimization, parameter estimation, and communication networks are described. Issues of asymptotic convergence and agreement are explored under very weak assumptions on the ordering of computations and the timing of information reception. Progress towards constructing a broadly applicable theory is surveyed.

†The research of D.P. Bertsekas was supported by NSF-ECS-8217668 and under DARPA Grant ONR-N00014-75-C-1183. The research of J.N. Tsitsiklis and M. Athans was supported by ONR-N00014-77-C-0532(NR041-519) and ONR N00014-84-K-0519.

*Dept. of Electrical Engineering and Computer Science, Laboratory for Information and Decision Systems, M.I.T., Cambridge, MA. 02139.

1. Introduction

Classical (centralized) theories of decision making and computation deal with the situation in which a single decision maker (man or machine) possesses (or collects) all available information related to a certain system and has to perform some computations and/or make a decision so as to achieve a certain objective. In mathematical terms, the decision problem is usually expressed as a problem of choosing a decision function that transforms elements of the information space into elements of the decision space so as to minimize a cost function. From the point of view of the theory of computation, we are faced with the problem of designing a serial algorithm which actually computes the desired decision.

Many real world systems however, such as power systems, communication networks, large manufacturing systems, C^3 systems, public or business organizations, are too large for the classical model of decision making to be applicable. There may be a multitude of decision makers (or processors), none of which possesses all relevant knowledge because this is impractical, inconvenient, or expensive due to limitations of the system's communication channels, memory, or computation and information processing capabilities.

In other cases the designer may deliberately introduce multiple processors into a system in view of the potential significant advantages offered by distributed computation. For problems where processing speed is a major bottleneck distributed computing systems may offer increases in throughput that are either unattainable or prohibitively expensive using a single processor. For problems where reliability or survivability is a major concern, distributed systems can offer increased fault tolerance or more graceful performance degradation in the face of various kinds of equipment failures. Finally as the cost of computation has decreased dramatically relative to the cost of communication it is now advantageous to trade off increased computation for reduced communication. Thus in database or sensor systems involving geographically separated data collection points it may be advantageous to process data locally at the point of collection and send condensed summaries to other points as needed rather than communicate the raw data to a single processing center.

For these reasons, we will be interested in schemes for distributed decision making and computation in which a set of processors (or decision makers) eventually compute a desired solution through a process of information exchange. It is possible to formulate mathematically a distributed decision problem whereby one tries to choose an "optimal" distributed scheme, subject to certain limitations. For example, we may impose constraints on the amount of information that may be transferred and look for a scheme which results in the best achievable decisions, given these constraints. Such problems have been formulated and studied in the decentralized control context [21,22], as well as in the computer science literature [23,24]. However, in practice these turn out to be very difficult, usually intractable problems [25,26]. We, therefore, choose to focus on distributed algorithms with a prespecified structure (rather than try to find an optimal structure): we assume that each processor chooses an initial decision and iteratively improves this decision as more information is obtained from the environment or other processors. By this we mean that the ith processor updates from time to time his decision x^i using some formula

$$x^i \leftarrow f^i(x^i, I^i) \qquad (1.1)$$

where I^i is the information available to the ith processor at the time of the update. In general there are serious limitations to this approach the most obvious of which is that the function f^i in (1.1) has to be chosen a priori on the basis of ad hoc considerations. However there are situations where the choice of reasonable functions f^i is not too dificult, and iterations such as (1.1) can provide a practical approach to an otherwise very difficult problem. After all, centralized counterparts of processes such as (1.1) are of basic importance in the study of stability of dynamic systems, and deterministic and stochastic optimization algorithms.

In most situations to be considered the information I^i of processor i contains some past decisions of other processors. However, we allow the possibility that some processors perform computations (using (1.1)) more often than they exchange information, in which case the information I^i may be outdated. Thus our formulation includes asynchronous algorithms where there is no strict a priori sequence according to which the iterations (1.1) are carried out at the various processors. Asynchronous algorithms have several advantages over their synchronous counterparts. First, while considerable progress has been made recently in understanding and reducing the computation and communication complexity of algorithm synchronization [42], the associated protocols still may require complex implementation and considerable communication and computation overhead. Second in a synchronous algorithm the progress of computation is controlled by the slowest processor. Finally, in situations where the problem data changes with time, synchronous algorithms require a restart protocol that may complicate their implementation and introduce critical time delays. On the other hand synchronous iterative algorithms are easier to understand and their convergence can be more readily established than their asynchronous counterparts. Some well known iterative algorithms simply do not converge or otherwise work satisfactorily when implemented in a totally asynchronous mode as will be explained in the sequel. Furthermore, the effects of asynchronism on rate of convergence and communication complexity are not well understood at present.

There are a number of characteristics and issues relating to the distributed iterative process (1.1) that either do not arise in connection with its centralized counterpart or else appear in milder form. First there is a graph structure characterizing the interprocessor flow of information. Second there is an expanded notion of the state of computation characterized by the current results of computation x^i and the latest information I^i available at the entire collection of processors i. Finally when (as we assume in this paper) there is no strict sequence according to which computation and communication takes place at the various processors the state of computation tends to evolve according to a point-to-set mapping and possibly in a probabilistic manner since each state of computation may give rise to many other states depending on which of the processors executes iteration (1.1) next and depending on possibly random exogenous information made available at the processors during execution of the algorithm.

From the point of view of applications, we can see several possible (broadly defined) areas. We discuss below some of them, although this is not meant to be an exhaustive list.

a) Parallel computing systems, possibly designed for a special purpose, e.g. for solving large scale mathematical programming problems with a particular structure. An important distinguishing feature of such systems is that the machine architecture is usually under the control of the designer. As mentioned above, we will assume a prespecified structure, thereby bypassing issues of architectural choice. However, the work surveyed in this paper can be useful for assessing the effects of communication delays and of the lack of synchronization in some parallel computing systems. Some of the early work on the subject [10],[11] is motivated by such systems. For a discussion of related issues see [7].

b) Data Communication Networks. Real time data network operation lends itself naturally to application of distributed algorithms. The structure needed for distributed computation (geographically distributed processors connected by communication links) is an inherent part of the system. Information such as link message flows, origin to destination data rates, and link and node failures is collected at geographically distributed points in the network. It is generally difficult to implement centralized algorithms whereby a single node would collect all information needed, make decisions, and transmit decisions back to the points of interest. The amount of data processing required of the central node may be too large. In addition the links over which information is transmitted to and from the central node are subject to failure thereby compounding the difficulties. For these reasons in many networks (e.g. the ARPANET) algorithms such as routing, flow control, and failure recovery are carried out in distributed fashion [1]-[5]. Since maintaining synchronization in a large data network generally poses implementation difficulties these algorithms are often operated asynchronously.

c) Distributed Sensor Networks and Signal Processing.

Suppose that a set of sensors obtain noisy measurements of a stochastic signal and then exchange messages with the purpose of computing a final estimate or identifying some unknown parameters. We are then interested in a scheme by which satisfactory estimates are produced without requiring that each sensor communicates his detailed information to a central processor. Some approaches that have been tried in this context may be found in [27,28,29,30].

d) Large Decentralized Systems and Organizations. There has been much interest, particularly in economics, in situations in which a set of rational decision makers make decisions and then update them on the basis of new information. Arrow and Hurwicz [31] have suggested a parallelism between the operation of an economic market and distributed computation. In this context the study of distributed algorithms may be viewed as an effort to model collective behavior. Similar models have been proposed for biological systems, [32]. Alternatively, finding good distributed algorithms and studying their communication requirements may yield insights on good ways of designing large organizations. It should be pointed out that there is an open debate concerning the degree of rationality that may be assumed for human decision makers. Given the cognitive limitations of humans, it is fair to say that only relatively simple algorithms can be meaningful in such contexts. The algorithms considered in this paper tend to be simple particularly when compared with other algorithms where decision makers attempt to process optimally the available information.

There are several broad methodological issues associated with iterative distributed algorithms such as correctness, computation or communication efficiency, and robustness. In this paper we will focus on two issues that generally relate to the question of validity of an algorithm.

a) Under what conditions is it possible to guarantee asymptotic convergence of the iterates x^i for all processors i, and asymptotic agreement between different processors i and j $[(x^i-x^j)\to 0]$?

b) How much synchronization between processor computations is needed in order to guarantee asymptotic convergence or agreement?

Significant progress has been made recently towards understanding these issues and the main purpose of this paper is to survey this work. On the other hand little is known at present regarding issues such as speed of convergence, and assessment of the value of communicated information in a distributed context. As a result we will not touch upon these topics in the present paper. Moreover, there are certain settings (e.g., decentralized control of dynamical systems, dynamic routing in data networks) in which issues of asymptotic convergence and agreement do not arise. Consequently, the work surveyed here is not of direct relevance to such situations.

In the next two sections we formulate a model of distributed asynchronous iterative computation, and illustrate its relevance by means of a variety of examples from optimization, parameter estimation, and communication networks. The model bears similarity to models of chaotic relaxation and distributed asynchronous fixed point computation [10]-[13] but is more general in two respects. First we allow two or more processors to update separately estimates of the same coordinate of the decision vector and combine their individual estimates by taking convex combinations, or otherwise. Second we allow processors to receive possibly stochastic measurements from the environment which may depend in nonlinear fashion on estimates of other processors. These generalizations broaden a great deal the range of applicability of the model over earlier formulations.

In Sections 4 and 5 we discuss two distinct approaches for analyzing algorithmic convergence. The first approach is essentially a generalization of the Lyapounov function method for proving convergence of centralized iterative processes. The second approach is based on the idea that if the processors communicate fast relative to the speed of convergence of computation then their solution estimates will be close to the path of a certain centralized process. By analyzing the convergence of this latter process one can draw inferences about the convergence of the distributed process. In Section 5 we present results related primarily to deterministic and stochastic descent optimization algorithms. An analysis that parallels Ljung's ODE approach [37],[38] to recursive stochastic algorithms may be found in [35] and in a forthcoming publication. In Section 6 we discuss convergence and agreement results for a special class of distributed processes in which the update of each processor, at any given time, is the optimal estimate of a solution given his information, in the sense that it minimizes the conditional expectation of a common cost function.

2. A Distributed Iterative Computation Model

In our model we are given a set of feasible decisions X and we are interested in finding an element of a special subset X* called the solution set. We do not specify X* further for the time being. An element of X* will be referred to as a solution. Without loss of generality we index all events of interest (message transmissions and receptions, obtaining measurements, performing computations) by an integer time variable t. This variable corresponds to a global clock and is mainly needed for analysis purposes. There is a finite collection of processors i=1,...,n each of which maintains an estimate $x^i(t) \in X$ of a solution and updates it once in a while according to a scheme to be described shortly. We do not assume that these processors have access to the global clock because this would amount to a synchronization assumption.

The ith processor receives from time to time m_i different types of measurements and maintains the latest values $z_1^i, z_2^i, \ldots, z_{m_i}^i$ of these measurements. The vector of measurements maintained by processor i is denoted by $z^i(t)$ and is an element of Z^i, the Cartesian product of sets Z_j^i, $j=1,\ldots,m_i$, i.e.

$$z^i(t) = (z_1^i(t),\ldots,z_{m_i}^i(t)) \in Z^i = Z_1^i \times \ldots \times Z_{m_i}^i \ .$$

We denote by T_j^i the set of times that processor i receives a new measurement of type j. It follows that

$$z_j^i(t) = z_j^i(t-1), \quad t \notin T_j^i \ .$$

At each time $t \in T_j^i$, processor i records the new value of z_j^i but also updates his estimate according to

$$x^i(t+1) = M_{ij}(x^i(t), z^i(t)), \quad t \in T_j^i, \tag{2.1}$$

where M_{ij} is a given function. A simple and important special case is when no update takes place upon reception of a measurement, in which case $M_{ij}(x,z)=x$, $\forall x,z$. Each processor i also updates from time to time (the set of such times is denoted by T^i) the estimate x^i according to

$$x^i(t+1) = C_i(x^i(t), z^i(t)), \quad t \in T^i, \tag{2.2}$$

where C_i is a given function. For all other times the estimate x^i remains unaffected, i.e.

$$x^i(t+1) = x^i(t), \qquad t \notin T^i.$$

For each processor i the sets T^i and T^i_j, $j=1,\dots,m_i$ are assumed mutually disjoint. As a practical matter this does not involve loss of generality. Thus <u>at each time t each processor i either receives a new measurement of type j and updates x^i according to (2.1), or updates x^i according to (2.2), or remains idle</u> in which case $x^i(t+1) = x^i(t)$ and $z^i(t) = z^i(t-1)$. The sequence according to which a processor executes (2.1) or (2.2) or remains idle (i.e. the sets T^i_j and T^i) is left unspecified and indeed much of the analysis in this paper is oriented towards the case where there is considerable a priori uncertainty regarding this sequence. Note that neither mapping M_{ij} or C_i involves a dependence on the time argument t. This is appropriate since it would be too restrictive to assume that all processors have access to a global clock. On the other hand the mappings M_{ij} and C_i may include dependences on local clocks (or counters) that record the number of times iterations (2.1) or (2.2) are executed at processor i. The value of the local counter of processor i may be artificially lumped as an additional component into the estimate x^i and incremented each time (2.1) or (2.2) are executed.

Note that there is redundancy in introducing the update formula (2.2) in addition to (2.1). We could view (2.2) as a special case of (2.1) corresponding to an update in response to a "self-generated" measurement at node i. Indeed such a formulation may be appropriate in some problems. On the other hand there is often some conceptual value in separating the types of updates at a processor in updates that incorporate new exogenous information (cf. (2.1)), and updates that utilize the existing information to improve the processor's estimate (cf. (2.2)).

The measurement $z_j^i(t)$, maintained by processor i at time t, is related to the processor estimates $x^1, x^2, \ldots, x^n$ according to an equation of the form

$$z_j^i(t) = \phi_{ij}(x^1(\tau_j^{i1}(t)), x^2(\tau_j^{i2}(t)), \ldots, x^n(\tau_j^{in}(t)), \omega), \qquad (2.3)$$

where ω belongs to the sample space Ω corresponding to a probability space (Ω, F, P), and $1 \leq \tau_j^{ik}(t) \leq t$, for every i,j,k.

We allow the presence of delays in equation (2.3) in the sense that the estimates $x^1, \ldots, x^n$ may be the ones generated via (2.1) or (2.2) at the corresponding processors at some times $\tau_j^{ik}(t)$, prior to the time t at which $z_j^i(t)$ is used by processor i. Furthermore the delays may be different for different processors. We place the following restriction on these delays which essentially guarantees that outdated information will eventually be purged from the system.

Assumption 2.1: For all i,j and k

$$\lim_{t\to\infty} \tau_j^{ik}(t) = \infty.$$

For the time being, the only other assumption regarding the timing and sequencing of measurement reception and estimate generation is the following:

Assumption 2.2: (Continuing update Assumption)

The sets T_j^i and T^i are infinite for any $i=1,\ldots,n$ and $j=1,\ldots,m_i$.

The assumption essentially states that each processor will continue to receive measurements in the future and update his estimate according to (2.1) and (2.2). Given that we are interested in asymptotic results there isn't much we can hope to prove without an assumption of this type. In order to formulate substantive convergence results we will also need further assumptions on the nature of the mappings M_{ij}, C_i, and ϕ_{ij} and possibly on the relative timing of measurement receptions, estimate updates and delays in (2.3) and these will be introduced later. In the next section we illustrate the model and its potential uses by means of examples.

It should be pointed out here that the above model is very broad and may capture a large variety of different situations, provided that the measurements z_j^i are given appropriate interpretations. For example, the choice $z_j^i(t) = x^j(\tau_j^{ij}(t))$ corresponds to a situation where processor i receives a message with the estimate computed by processor j at time $\tau_j^{ij}(t)$, and $t-\tau_j^{ij}(t)$ may be viewed as a communication delay. In this case processors act also as sensors generating measurements for other processors. In other situations however specialized sensors may generate (possibly noisy and delayed) feedback to the processors regarding estimates of other processors (cf. (2.3)). Examples of both of these situations will be given in the next section.

3. Examples

An important special case of the model of the previous section is when the feasible set X is the Cartesian product of n sets

$$X = X_1 \times X_2 \times \ldots \times X_n ,$$

each processor i is assigned the responsibility of updating the ith component of the decision vector $x = (x_1,x_2,\ldots,x_n)$ via (2.2) while receiving from each processor j ($j \neq i$) the value of the jth component x_j. We refer to such distributed processes as being <u>specialized</u>. The first five examples are of this type.

Example 1: (Shortest Path Computation)

Let (N,A) be a directed graph with set of nodes $N=\{1,2,\ldots,n\}$ and set of links A. Let N(i) denote the set of downstream neighbors of node i, i.e. the nodes j such that (i,j) is a link. Assume that each link (i,j) is assigned a positive scalar a_{ij} referred to as its length. Assume also that there is a directed path to node 1 from every other node. Let x_i^i be the estimate of the shortest distance from node i to node 1 available at node i. Consider a distributed algorithm whereby each node i=2,...,n executes the iteration

$$x_i^i \leftarrow \min_{j\in N(i)} \{a_{ij}+x_j^j\} \tag{3.1}$$

after receiving one or more estimates x_j^j from its neighbors, while node 1 sets

$$x_1^1 = 0.$$

This algorithm--a distributed asynchronous implementation of Bellman's shortest path algorithm--was implemented on the ARPANET in 1969 [14], and subsequently in other computer networks. (Actually in the version implemented on the ARPANET the lengths a_{ij} were allowed to change with time at a fast rate and this created serious algorithmic difficulties [14]). The estimate x_i^i can be shown to converge to the unique shortest distance from node i to node 1 provided the starting values x_i^i are nonnegative [12]. The algorithm clearly is a special case of the model of the previous section. Here the measurement equation [cf. (2.3)] is

$$z_j^i(t) = x_j^j(\tau_j^{ij}(t)), \quad t\in T_j^i,\ j\in N(i) \ ; \tag{3.2}$$

also, measurement receptions [cf. (2.1)] leave x^i unchanged. The update formula corresponding to (2.2) is based on (3.1). Its ith coordinate is given by

$$x_i^i(t) = \min_{j\in N(i)} \{a_{ij}+z_j^i(t)\}, \qquad t\in T^i, \tag{3.3}$$

while for the remaining coordinates its form is not material (processor i is concerned in effect only with the shortest distance from node i to node 1).

Example 2: (Fixed point calculations)

The preceding example is a special case of a distributed dynamic programming algorithm (see [12]) which is itself a special case of a distributed fixed point algorithm. Suppose we are interested in computing a fixed point of a mapping $F: X \to X$. We construct a distributed fixed point algorithm that is a special case of the model of the previous section as follows:

Let X be a Cartesian product of the form $X = X_1 x X_2 \; x \ldots x X_n$ and let us write accordingly $x = (x_1, x_2, \ldots, x_n)$ and $F(x) = (F_1(x), F_2(x), \ldots, F_n(x))$ where $F_i: X \to X_i$. Let $x^i = (x_1^i, \ldots, x_n^i)$ be the estimate of x generated at the ith processor. Processor i executes the iteration

$$x_j^i(t+1) = \begin{cases} z_j^i(t) & \text{if } t \in T_j^i, \; i \neq j \\ F_i(x^i(t)) & \text{if } t \in T^i, \; i = j \\ x_j^i(t) & \text{otherwise} \end{cases} \tag{3.4}$$

(this iteration defines both mappings C_i of (2.2) and M_{ij} of (2.1)) and transmits from time to time x_i^i to the other processors. Thus the measurements z_j^i are given by [cf. (2.3)].

$$z_j^i(t) = x_j^j(\tau_j^{ij}(t)), \quad i \neq j \; . \tag{3.5}$$

Conditions under which the estimate x^i converges to a fixed point of F are given in [13] (see also Section 4).

<u>Example 3</u>: (Distributed deterministic gradient algorithm)

This example is a special case of the preceding one whereby $X = R^n$, $X_i = R$, and F is of the form

$$F(x) = x - \alpha\nabla f(x) \tag{3.6}$$

where ∇f is the gradient of a function $f: R^n \to R$, and α is a positive scalar stepsize. Iteration (3.4) can then be written as

$$x_j^i(t+1) = \begin{cases} z_j^i(t) & \text{if } t\in T_j^i,\ i\neq j \\ x_i^i(t) - \alpha \dfrac{\partial f(x^i(t))}{\partial x_i} & \text{if } t\in T^i,\ i=j \\ x_j^i(t) \quad \text{otherwise} \end{cases} \tag{3.7}$$

A variation of this example is obtained if we assume that, instead of each processor i transmitting directly his current value of the coordinate x_i to the other processors, there is a measurement device that transmits a value of the partial derivative $\frac{\partial f}{\partial x_i}$ to the ith processor. In this case there is only one type of measurement for each processor i [cf. (2.3)] and it is given by

$$z_1^i(t) = \frac{\partial f[x_1^1(\tau_1^{i1}(t)),\ldots,x_n^n(\tau_1^{in}(t))]}{\partial x_i} .$$

Iteration (3.7) takes the form

$$x_i^i(t+1) = x_i^i(t) - \alpha z_1^i(t) \qquad \text{if } t\in T^i,$$

for the ith coordinate, while for the remaining coordinates its form is immaterial (processor i in effect estimates only the ith coordinate x_i^i). The equations above assume no noise in the measurement of each partial derivative; one could also consider the situation where this measurement is corrupted by additive or multiplicative noise thereby obtaining a model of a distributed stochastic gradient method. Many other descent algorithms admit a similar distributed version.

Example 4: (An Organizational Model)

This example is a variation of the previous one, but may be also viewed as a model of collective decision making in a large organization. Let $X = X_1 x X_2 x \ldots x X_n$ be the feasible set, where X_i is a Euclidean space and let $f: X \to [0,\infty)$ be a cost function of the form $f(x) = \sum_{i=1}^{n} f^i(x)$. We interpret f^i as the cost facing the i-th division of an organization. This division is under the authority of decision maker (processor) i, who updates the i-th component $x_i \in X_i$ of the decision vector x. We allow the cost f^i to depend on the decisions x_j of the remaining decision makers, but we assume that this dependence is weak. That is, let

$$K^i_{jm} = \sup_{x \in X} \left| \frac{\partial^2 f^i(x)}{\partial x_j \partial x_m} \right|$$

and we are interested in the case $K^i_{jm} << K^i_{ii}$ (unless j=m=i). Decision maker i receives measurements z^i_j, j=1,...,n of the form

$$z^i_j(t) = \frac{\partial f^j}{\partial x_i}(x^1_1(\tau^{i1}_j(t)), x^2_2(\tau^{i2}_j(t)), \ldots, x^n_n(\tau^{in}_j(t))) , \qquad (3.8)$$

where $\tau^{ik}_j(t) \leq t$ [cf. (2.3)]. Once in a while, he also updates his decision according to

$$x^i_i(t+1) = x^i_i(t) - \alpha_i \sum_{j=1}^{n} z^i_j(t) , \quad t \in T^i. \qquad (3.9)$$

If we assume that

$$\tau^{im}_j(t) \leq \tau^{ij}_j(t), \qquad \forall i,j,m,t,$$

the above algorithm admits the following interpretation: each decision maker m, at time $\tau_j^{im}(t)$ sends a message $x_m^m(\tau_j^{im}(t))$, to inform decison maker j of his decision. Then, at time $\tau_j^{ij}(t)$, decision maker j (who is assumed to be knowledgeable about f^j) computes z_j^i according to (3.8) and sends it to decision maker i who, in turn, uses it to update his decision according to (3.9). On an abstract level, each decision maker j is being informed about the decision of the others and replies by saying how he is affected by their decisions; however, this may be done in an asynchronous and very irregular manner.

Example 5: (Distributed optimal routing in data networks)

A standard model of optimal routing in data networks (see e.g. the survey [6]) involves the multicommodity flow problem

$$\begin{aligned}
&\text{minimize } \sum_{a\varepsilon A} D_a(F_a) \\
&\text{subject to } F_a = \sum_{w\varepsilon W} \sum_{\substack{p\varepsilon P_w \\ a\varepsilon p}} x_{w,p}, \ \forall a\varepsilon A \\
&\qquad \sum_{p\varepsilon P_w} x_{w,p} = r_w, \qquad \forall w\varepsilon W \\
&\qquad x_{w,p} \geq 0, \qquad \forall w\varepsilon W,\ p\varepsilon P_w .
\end{aligned}$$

Here A is the set of directed links in a data network, F_a is the communication rate (say in bits/sec) on link $a\varepsilon A$, W is a set of origin-destination

(OD) pairs, P_w is a given set of directed paths joining the origin and the destination of OD pair w, $x_{w,p}$, is the communication rate on path p of OD pair w, r_w is a given required communication rate of OD pair w, and D_a is a monotonically increasing differentiable convex function for each $a \in A$. The objective here is to distribute the required rates r_w among the available paths in P_w so as to minimize a measure of average delay per message as expressed by $\sum_{a \in A} D_a(F_a)$.

Since the origin node of each OD pair w has control over the rates $x_{w,p}$, $p \in P_w$, it is convenient to use a distributed algorithm of the gradient projection type (see [6], [8], [46]), whereby each origin iterates on its own path rates asynchronously and independently of other origins. This type of iteration requires knowledge of the first partial derivatives $D_a'(F_a)$ for each link, evaluated at the current link rates F_a. A practical scheme similar to the one currently adopted on the ARPANET [9] is for each link $a \in A$ to broadcast to all the nodes the current value of either F_a or $D_a'(F_a)$. This information is then incorporated in the gradient projection iteration of the origin nodes. In this scheme each origin node can be viewed as a processor and F_a or $D_a'(F_a)$ plays the role of a measurement which depends on the solution estimates of all processors [cf. (2.3)].

The direct opposite of a specialized process, in terms of division of labor between processors, is a totally overlapping process.

Example 6: (Gradient Method; Total Overlap)

Let the feasible set X be a Euclidean space. Each processor i receives measurements z_j^i ($j \neq i$) which are the values of the estimates x^j of other processors; that is,

$$z_j^i(t) = x^j(\tau_j^{ij}(t)), \quad i \neq j \ .$$

Whenever such a measurement is received, processor i updates his estimate by taking a convex combination:

$$x^i(t+1) = M_{ij}(x^i(t), z_j^i(t)) = \beta^{ij} x^i(t) + (1-\beta^{ij}) z_j^i(t), \quad t \in T_j^i, \; i \neq j \tag{3.10}$$

where $0<\beta^{ij}<1$. Also processor i receives his own information z_i^i, generated according to

$$z_i^i(t) = \phi_{ii}(x^i(t),\omega) = \frac{\partial f(x^i(t))}{\partial x}$$

and updates x^i according to

$$x^i(t+1) = M_{ii}(x^i(t), z^i(t)) = x^i(t) - \alpha^i z_i^i(t), \quad t \in T_i^i \tag{3.11}$$

where α^i is a positive scalar stepsize.[†] Such an algorithm is of interest if the objective is to minimize a cost function $f: X \to \mathbb{R}$, and $z_i^i(t)$ is in some sense a descent direction with respect to f, e.g. as assumed above. In a deterministic setting, such a scheme could be redundant, as some processors would be close to replicating the computation of others. In a stochastic setting, however (e.g. if

$$z_i^i(t) = \frac{\partial f}{\partial x}(x^i(t)) + w^i(t),$$

where $w^i(t)$ is zero-mean white noise) the combining process is effectively averaging out the effects of the noise and may improve convergence .

Example 7: (System Identification)

Consider two stochastic processes $y^1(t), y^2(t)$ generated according to

$$y^i(t) = A(q)u(t) + w^i(t),$$

[†]The stepsize α^i could be constant as in deterministic gradient methods. However, in other cases (such as stochastic gradient methods with additive noise) it is essential that α^i is time varying and tends to zero. This, strictly speaking, violates the assumption that the mapping M_{ij} does not depend on the time t. However it is possible to circumvent this by introducing (as an additional component of x^i) a local counter at each processor i that keeps track of the number of times iteration (3.10) or (3.11) is executed at processor i. The stepsize α^i could be made dependent on the value of this local counter (see the discussion following (2.1) and (2.2) in Section 2).

where $A(\cdot)$ is a polynomial, to be identified, q is the unit delay operator and $w^i(t)$, $i=1,2$, are white, zero-mean processes, possibly correlated with each other. Let there be two processors ($n=2$); processor i measures $y^i(t)$ and both measure $u(t)$ at each time t. Each processor i updates his estimate x^i of the coefficients of A according to any of the standard system identification algorithms (e.g. the LMS or RLS algorithm). Under the usual identifiability conditions [33] each processor would be able to identify $A(\cdot)$ by himself. However, convergence should be faster if once in a while one processor gets (possibly delayed) measurements of the estimates of the other processor and combines them by taking a convex combination. Clearly this is a case of total overlap, as in Example 6.

A more complex situation arises if we have two ARMAX processes y^1, y^2, driven by a common colored noise $w(t)$:

$$A^i(q)y^i(t) = B^i(q)u^i(t) + w(t), \quad i=1,2$$
$$w(t) = C(q)v(t),$$

where $v(t)$ is white and A^i,B^i,C are polynomials in the delay operator q. Assuming that each processor i observes y^i and u^i, he may under certain conditions [34] identify A^i,B^i. In doing this he must, however, identify the common noise source C as well. So we may envisage a scheme whereby processor i uses a standard algorithm to identify A^i, B^i, C and once in a while receives messages with the other processor's estimates of the coefficients of C; these estimates are then combined by taking a convex combination.

This latter example falls in between the extreme cases of specialization and total overlap: there is specialization concerning the coefficients of A^i,B^i and overlap concerning the coefficients of C.

4. Convergence of Contracting Processes

In our effort to develop a general convergence result for the distributed algorithmic model of Section 2 we draw motivation from existing convergence theories for (centralized) iterative algorithms. There are several theories of this type (see Zangwill [15], Luenberger [16], Ortega and Rheinboldt [17], Polak, [18], Poljak [19]). Most of these theories have their origin in Lyapunov's stability theory for differential and difference equations. The main idea is to consider a generalized distance function (or Lyapunov function) of the typical iterate to the solution set. In optimization methods the objective function is often suitable for this purpose while in equation solving methods a norm of the difference between the current iterate and the solution is usually employed. The idea is typically to show that at each iteration the value of the distance function is reduced and reaches its minimum value in the limit.

The result of this section is based on a similar idea. However instead of working with a generalized distance function we prefer to work (essentially) with the level sets of such a function; and instead of working with a single processor iterate (as in centralized processes) we work with what may be viewed as a state of computation of the distributed process which includes all current processor iterates and all latest information available at the processors.

The starting point for development of our result is a nested sequence of subsets $\{X(k)\}$ of the feasible set X. We assume that

$$X^* \subset X(k+1) \subset X(k) \subset \cdots \subset X \quad . \tag{4.1}$$

The sequence $\{X(k)\}$ depends on the nature of the specific problem at hand but the implication is that <u>membership of an estimate x^i in the set $X(k)$ is representative of its</u>

proximity to the solution set X^*. This is formalized by assuming that if $\{x_k\}$ is a sequence in X such that $x_k \in X(k)$ for all k, then every limit point of $\{x_k\}$ belongs to the solution set X^*. We assume here that X is a topological space so we can talk about convergence of sequences in X. A relevant example is when X is R^n, x^* is a unique solution and $X(k) = \{x| \; ||x-x^*|| \leq B\alpha^k\}$, where B and α are constants with $B>0$ and $0<\alpha<1$, and $||\cdot||$ is some norm on R^n.

We denote for all i,j and k

$$Z^i_j(k) = \{\phi_{ij}(x^1,\ldots,x^n,\omega) | x^1 \varepsilon X(k),\ldots,x^n \varepsilon X(k), \quad \omega \varepsilon \Omega\} \tag{4.2}$$

$$Z^i(k) = Z^i_1(k) x Z^i_2(k) x \ldots x Z^i_{m_i}(k), \tag{4.3}$$

$$\overline{X}^i(k) = \{C_i(x^i,z^i) | x^i \varepsilon X(k), \; z^i \varepsilon Z^i(k)\} \tag{4.4}$$

$$\overline{Z}^i_j(k) = \{\phi_{ij}(x^1,\ldots,x^n,\omega) | x^1 \varepsilon \overline{X}^1(k),\ldots,x^n \varepsilon \overline{X}^n(k), \; \omega \varepsilon \Omega\}, \tag{4.5}$$

$$\overline{Z}^i(k) = \overline{Z}^i_1(k) x \overline{Z}^i_2(k) x \ldots x \overline{Z}^i_{m_i}(k). \tag{4.6}$$

In words, $\overline{X}^i(k)$ is the set of estimates obtained at a processor i after an update $x^i \leftarrow C_i(x^i,z^i)$ based on values of x^i and z^i that reflect membership of all processor estimates in the set X(k); $\overline{Z}^i(k)$ is the set of measurements obtained at processor i that reflect membership of the estimate of each processor j in the corresponding set $\overline{X}^j(k)$.

We consider the following two assumptions the first of which applies to the important special case where a measurement reception does not trigger an update of a processor's estimate.

Assumption 3.1: The measurement update iteration (2.1) leaves the estimate $x^i(t)$ unchanged. Furthermore for all i and k

$$\overline{X}^i(k) \subset X(k) \tag{4.7}$$

$$C_i(x^i,z^i) \in X(k+1), \quad \forall x^i \in \overline{X}^i(k),\ z^i \in \overline{Z}^i(k) . \tag{4.8}$$

A more general version is:

Assumption 3.1': The sets X(k) and the mappings ϕ_{ij}, M_{ij}, and C_i are such that for all i,j and k

$$\overline{X}^i(k) \subset X(k) \tag{4.9}$$

$$M_{ij}(x^i,z^i) \varepsilon X(k), \qquad \forall x^i \varepsilon X(k),\ z^i \varepsilon Z^i(k), \tag{4.10}$$

$$M_{ij}(x^i,z^i) \varepsilon \overline{X}^i(k), \qquad \forall x^i \varepsilon \overline{X}^i(k),\ z^i \varepsilon Z^i(k), \tag{4.11}$$

$$C_i(x^i,z^i) \varepsilon X(k+1), \qquad \forall x^i \varepsilon \overline{X}^i(k),\ z^i \varepsilon \overline{Z}^i(k), \tag{4.12}$$

$$M_{ij}(x^i,z^i) \varepsilon X(k+1), \qquad \forall x^i \varepsilon X(k+1),\ z^i \varepsilon \overline{Z}^i(k) . \tag{4.13}$$

As discussed earlier, if we can show that a processor's estimate successively moves from the set X(0) to X(1), then to X(2) and so on, then convergence to a solution is guaranteed. Assumption 3.1 or 3.1' guarantee that this will occur based on the following observations where for simplicity we assume that there are no communication delays i.e. $\tau^i_j(t) = 0$ for $t \varepsilon T^i_j$.

a) Once all processors' estimates move into X(k)(and the measurements available reflect that, i.e., $z^i \in Z^i(k)$ for all i), they will remain in X(k) [cf. the definition (4.4) and either (4.7) or (4.9), (4.10)]; and once any processor's i estimate moves into $\overline{X}^i(k)$ it will remain in $\overline{X}^i(k)$ [cf. (4.7) or (4.9)-(4.11)].

b) Once the information available at processor i reflects membership of every processor's j estimate in the corresponding set $\overline{X}^j(k)$ [i.e $x^i \in \overline{X}^i(k), z^i \in \overline{Z}^i(k)$], then an update $x^i \leftarrow C_i(x^i, z^i)$ results in an estimate x^i belonging to $X(k+1) \cap \overline{X}^i(k)$ [cf. (4.8) or (4.12)].

c) Finally once a processor's estimate moves into the set $X(k+1) \cap \overline{X}^i(k)$ it remains there in view of (4.8) or (4.11)-(4.13).

These observations prove in effect the following proposition:

Proposition 3.1: Let Assumptions 3.1 or 3.1' hold. Assume that all initial processor estimates $x^i(0)$, $i=1,\ldots,n$ belong to $X(0)$, while all initial measurements $z^i_j(0)$ available at the processors belong to the corresponding sets $Z^i_j(0)$ of (4.2). Then every limit point of the sequences $\{x^i(t)\}$ is almost surely a solution.

We note that Assumption 3.1 and 3.1' are generalized versions of a similar assumption in [13], and therefore Proposition 3.1 is a stronger version of the result given in that reference. Note also that the assumptions do not differentiate the effects of two different members of the probability space, so they apply to situations where the process is either deterministic (Ω consists of a single element), or else stochastic variations are not sufficiently pronounced to affect the membership relations in (4.7)-(4.13). We now provide some examples drawn primarily from problems of gradient optimization and solution of nonlinear equations. Applications in dynamic programming are described in [12].

Example 2 (continued): Consider the specialized process for computing a fixed point of a mapping F in example 2. There X is a Cartesian product $X_1 x X_2 \; x \ldots x \; X_n$, and each processor i is responsible for updating the ith coordinate x_i of $x=(x_1,x_2,\ldots,x_n)$

while relying on essentially direct communications from other processors to obtain estimates of the other coordinates. Suppose that each set X_i is a Euclidean space with norm $||\cdot||_i$ and X is endowed with the sup norm

$$||x|| = \max \{||x_1||_1,\dots,||x_n||_n\}, \quad \forall x\varepsilon X. \tag{4.14}$$

Assume further that F is a contraction mapping with respect to this norm, i.e., for some $\alpha\varepsilon(0,1)$

$$||F(x)-F(y)|| \leq \alpha||x-y||, \quad \forall x,y\varepsilon X . \tag{4.15}$$

Then the solution set consists of the unique fixed point x* of F. For some positive constant B let us consider the sequence of sets

$$X(k) = \{x\varepsilon X|\ ||x-x^*|| \leq B\alpha^k\}, \qquad k=0,1,\dots$$

The sets defined by (4.2)-(4.6) are then given by

$$Z_j^i(k) = \{x_j\in X_j|\ ||x_j-x_j^*||_j \leq B\alpha^k\}, \quad \forall j\neq i,$$

$$\overline{X}^i(k) = \{x\varepsilon X(k)|\ ||x_i-x_i^*||_i \leq B\alpha^{k+1}\} ,$$

$$\overline{Z}_j^i(k) = \{x_j\in X_j|\ ||x_j-x_j^*||_j \leq B\alpha^{k+1}\} , \quad \forall j\neq i$$

It is straightforward to show that the sequence {X(k)} satisfies Assumption 3.1'. (Assumption 3.1 would be satisfied if an inconsequential modification is introduced in the model and, in place of (3.4), a measurement reception leaves the processor's estimate unchanged while computation updates at times $t\in T^i$ have the form

$$x_j^i(t) = \begin{cases} z_j^i(t) & \text{if } i\neq j \\ F_i(z_1^i(t),\dots,z_{i-1}^i(t),x_i^i(t),z_{i+1}^i(t),\dots,z_n^i(t)), & \text{if } i=j). \end{cases}$$

Further illustrations related to this example are given in [13]. Note however that the use of the sup norm (4.14) is essential for the verification of Assumption 3.1'.

Similarly Assumption 3.1' can be verified in the preceding example if the contraction assumption is substituted by a monotonicity assumption (see [13]). This monotonicity assumption is satisfied by most of the dynamic programming problems of interest including the shortest path problem of example 1 (see also [12]). An important exception is the infinite horizon average cost Markovian decision problem (see [12], p. 616). Another interesting application of Proposition 3.1 to analysis of asynchronous flow control algorithms in data networks can be found in Mosely [45].

An important special case for which the contraction mapping assumption (4.15) is satisfied arises when $X=R^n$ and $x_1,x_2,\ldots,x_n$ are the coordinates of x. Suppose that F satisfies

$$|F(x)-F(y)|\leq P|x-y|, \qquad \forall x,y\varepsilon R^n$$

where P is an nxn matrix with nonnegative elements and spectral radius strictly less than unity, and for any $z=(z_1,z_2,\ldots,z_n)$ we denote by $|z|$ the column vector with coordinates $|z_1|,|z_2|,\ldots,|z_n|$. Then F is called a P-contraction mapping. Fixed point problems involving such mappings arise in dynamic programming ([20], p. 374), and solution of systems of nonlinear equations ([17], Section 13.1). It can be shown ([11], p.231) that if F is a P-contraction then it is a contraction mapping with respect to some norm of the form (4.14). Therefore Proposition 3.1 applies.

We finally note that it is possible to use Proposition 3.1 to show convergence of similar fixed point distributed processes involving partial or total overlaps between the processors (compare with example 6).

Example 3 (continued): Consider the special case of the deterministic gradient algorithm of example 3 corresponding to the mapping

$$F(x) = x - \alpha\nabla f(x) \ . \tag{4.16}$$

Assume that $f: R^n \to R$ is a twice continuously differentiable convex function with Hessian matrix $\nabla^2 f(x)$ which is positive definite for all x. Assume also that there exists a unique minimizing point x* of f over R^n. Consider the matrix

$$H^* = \begin{bmatrix} \left|\frac{\partial^2 f}{(\partial x_1)^2}\right|, & -\left|\frac{\partial^2 f}{\partial x_1 \partial x_2}\right| & ,\ldots\ldots, & -\left|\frac{\partial^2 f}{\partial x_1 \partial x_n}\right| \\ \vdots & \vdots & & \vdots \\ -\left|\frac{\partial^2 f}{\partial x_n \partial x_1}\right|, & -\left|\frac{\partial^2 f}{\partial x_n \partial x_2}\right| & ,\ldots\ldots, & \left|\frac{\partial^2 f}{(\partial x_n)^2}\right| \end{bmatrix} \tag{4.17}$$

obtained from the Hessian matrix $\nabla^2 f(x^*)$ by replacing the off-diagonal terms by their negative absolute values. It is shown in [13] that if the matrix H* is positive definite then the mapping F of (4.16) is a P-contraction within some open sphere centered at x* provided the stepsize α in (4.16) is sufficiently small. Under these circumstances the distributed asynchronous gradient method of this example is convergent to x* provided all initial processor estimates are sufficiently close to x* and the stepsize α is sufficiently small. The neighborhood of local convergence will be larger if the matrix (4.17) is positive definite within an accordingly larger neighborhood of x*. For example if f is positive definite quadratic with the corresponding matrix (4.17) positive definite a global convergence result can be shown.

One condition that guarantees that H* is positive definite is strict diagonal dominance ([17], p. 48-51)

$$\frac{\partial^2 f}{(\partial x_i)^2} > \sum_{\substack{j=1 \\ j\neq i}}^{n} \left| \frac{\partial^2 f}{\partial x_i \partial x_j} \right| , \qquad \forall i=1,\ldots,n,$$

where the derivatives above are evaluated at x*. This type of condition is typically associated with situations where the coordinates of x are weakly coupled in the sense that changes in one coordinate have small effects on the first partial derivatives of f with respect to the other coordinates. This result can be generalized to the case of weakly coupled systems (as opposed to weakly coupled coordinates). Assume that x is partitioned as $x=(x_1,x_2,\ldots,x_n)$ where now $x_i \in R^{m_i}$ (m_i may be greater than one but all other assumptions made earlier regarding f are in effect). Assume that there are n processors and the ith processor asynchronously updates the subvector x_i according to an approximate form of Newton's method where the second order submatrices of the Hessian $\nabla^2_{x_i x_j} f$, $i\neq j$ are neglected, i.e.

$$x_i \leftarrow x_i - (\nabla^2_{x_i x_i} f)^{-1} \nabla_{x_i} f . \tag{4.18}$$

In calculating the partial derivatives above processor i uses the values x_j latest communicated from the other processors $j\neq i$ similarly with the distributed gradient method. It can be shown that if the cross-Hessians $\nabla^2_{x_i x_j} f$, $i\neq j$ have sufficiently small norm relative to $\nabla^2_{x_i x_i} f$, then the totally asynchronous version of the approximate Newton method (4.18) converges to x* if all initial processor estimates are sufficiently close to x*. The same type of result may also be shown if (4.18) is replaced by

$$x_i \leftarrow \arg \min_{x_i \in R^{m_i}} f(x_1, x_2, \ldots, x_n) \,. \tag{4.19}$$

Unfortunately it is not true always that the matrix (4.17) is positive definite, and there are problems where coupling between coordinates is strong and the totally asynchronous version of the distributed gradient method is not guaranteed to converge regardless of how small the stepsize α is chosen. The following example demonstrates the nature of the difficulty.

Counterexample: Consider the function $f: R^3 \to R$ given by

$$f(x_1, x_2, x_3) = (x_1 + x_2 + x_3)^2 + (x_1 + x_2 + x_3 - 3)^2 + \varepsilon(x_1^2 + x_2^2 + x_3^2)$$

where $0 < \varepsilon << 1$. The optimal solution is close to $(\frac{1}{2}, \frac{1}{2}, \frac{1}{2})$ for ε small. The scalar ε plays no essential role in this example. It is introduced merely for the purpose of making f positive definite. Assume that all initial processor estimates are equal to some common value $\bar{x}$, and that processors execute many gradient iterations with a small stepsize before communicating the current values of their respective coordinates to other processors. Then (neglecting the terms that depend on ε) the ith processor tries in effect to solve the problem

$$\min_{x_i} \{(x_i + 2\bar{x})^2 + (x_i + 2\bar{x} - 3)^2\}$$

thereby obtaining a value close to $\frac{3}{2} - 2\bar{x}$. After the processor estimates of the respective coordinates are exchanged each processor coordinate will have been updated approximately according to

$$\bar{x} \leftarrow \frac{3}{2} - 2\bar{x} \tag{4.20}$$

and the process will be repeated. Since (4.20) is a divergent iterative process we see that, regardless of the stepsize chosen and the proximity of the initial processor estimates to the optimal solution, by choosing the delays between successive communications sufficiently large the distributed gradient method can be made to diverge when the matrix H* of (4.17) is not positive definite.

5. Convergence of Descent Processes

We saw in the last section that the distributed gradient algorithm converges appropriately when the matrix (4.17) is positive definite. This assumption is not always satisfied, but convergence can be still shown (for a far wider class of algorithms) if a few additional conditions are imposed on the frequency of obtaining measurements and on the magnitude of the delays in equation (2.3). The main idea behind the results described in this section is that if delays are not too large, if certain processors do not obtain measurements and do not update much more frequently than others, then the effects of asynchronism are relatively small and the algorithm behaves approximately as a centralized algorithm.

As an illustration consider the deterministic gradient method (4.16) for the case where the objective f is quadratic and there are two variables and two processors each specializing in updating a single coordinate of the solution. Each processor i executes the iteration

$$x_i^i \leftarrow x_i^i - \alpha \frac{\partial f}{\partial x_i}, \qquad i=1,2 \tag{5.1}$$

and occasionally communicates the latest value of x_i^i to the other processor.

The partial derivative $\frac{\partial f}{\partial x_i}$ is evaluated at the current estimate $x^i = (x_1^i, x_2^i)$ of the processor. Let us try to estimate the effect of communication delays by assuming identical initial processor estimates $x^i = (x_1, x_2)$ and two sequences of events. In the first sequence processor 1 executes (5.1), communicates the result to processor 2 who then executes (5.1) and communicates the result to processor 1. The final estimates at the two processors will then be

$$\bar{x}^i = (\bar{x}_1, \bar{x}_2) , \qquad i=1,2$$

where

$$\bar{x}_1 = x_1 - \alpha \frac{\partial f(x_1,x_2)}{\partial x_1} \tag{5.2}$$

$$\bar{x}_2 = x_2 - \alpha \frac{\partial f(\bar{x}_1,x_2)}{\partial x_2} \tag{5.3}$$

Since f is quadratic we have

$$\frac{\partial f(\bar{x}_1,x_2)}{\partial x_2} = \frac{\partial f(x_1,x_2)}{\partial x_2} + \frac{\partial^2 f(x_1,x_2)}{\partial x_1 \partial x_2} (\bar{x}_1 - x_1) ,$$

so using (5.2) and (5.3) we obtain

$$\bar{x}_2 = x_2 - \alpha \frac{\partial f(x_1,x_2)}{\partial x_2} - \alpha^2 \frac{\partial^2 f(x_1,x_2)}{\partial x_1 \, \partial x_2} \frac{\partial f(x_1,x_2)}{\partial x_1} \tag{5.4}$$

In the second sequence of events there is a one unit communication delay between execution of (5.1) at processor 1 and communication to processor 2, so that processor 2 executes the iteration (5.1) without knowledge of the latest estimate of processor 1. After exchange of the results of the updates the final estimates at both processors will be

$$\tilde{x}^i = (\tilde{x}_1, \tilde{x}_2)$$

where

$$\tilde{x}_1 = x_1 - \alpha \frac{\partial f(x_1,x_2)}{\partial x_1} \tag{5.5}$$

$$\tilde{x}_2 = x_2 - \alpha \frac{\partial f(x_1,x_2)}{\partial x_2} \quad . \tag{5.6}$$

Comparing (5.2)-(5.6) we see that the effect of the communication delay is reflected in a difference of magnitude $\alpha^2 \frac{\partial^2 f(x_1,x_2)}{\partial x_1 \partial x_2} \frac{\partial f(x_1,x_2)}{\partial x_1}$ in the second coordinate. This difference is of <u>second order in the stepsize α,</u> so if α is small enough the effect of the communication delay is negligible. This example can be easily generalized to show that <u>the effect of any finite amount of communication delay is of second order in the stepsize</u> and can be made negligible by choosing α sufficiently small. This fact underlies the proofs of all the results presented in this section.

Consider now the general problem of unconstrained minimization of f: $R^n \to R$. We assume that f is convex, has Lipschitz continuous first derivatives and is bounded below. Consider the distributed asynchronous gradient method of Example 3, where each processor i=1,...,n specializes in updating x_i, the i-th coordinate of x. The algorithm is specified by equations (3.5) and (3.7) which are repeated below for easy reference

$$x_j^i(t+1) = \begin{cases} z_j^i(t) & \text{if } t\in T_j^i,\ i\neq j & (5.7a) \\ x_i^i(t) - \alpha \frac{\partial f(x^i(t))}{\partial x_i}, & \text{if } t\in T^i,\ i=j, & (5.7b) \\ x_j^i(t) & \text{otherwise} & (5.7c) \end{cases}$$

$$z_j^i(t) = x_j^j(\tau_j^{ij}(t)) \ , \quad i\neq j. \tag{5.8}$$

Proposition 5.1: Assume that, for some $B>0$, we have $t-\tau_j^{ij}(t)<B$, $\forall i,j,t$, and that the difference between consecutive elements of T^i is bounded, for each i. Then, there exists some $\bar{\alpha}>0$ such that, for all $\alpha\in(0,\bar{\alpha}]$, every limit point of $x^i(t)$, for any i, is a minimizing point of f.

Proof (Outline): Let $y(t) = (x_1^1(t),\dots,x_n^n(t))$. We also define a vector $z(t)\in R^n$ whose i-th component is $-\dfrac{\partial f(x^i(t))}{\partial x_i}$ if $t\in T^i$, zero otherwise. Then, (5.7b) yields

$$y(t+1) = y(t) + \alpha z(t) \tag{5.9}$$

Using the Lipschitz continuity of $\partial f/\partial x$ and (5.7),(5.8) we can easily show that

$$\left|\frac{\partial f}{\partial x_i}(x^i(t)) - \frac{\partial f}{\partial x_i}(y(t))\right| \le A_1\,\alpha \sum_{\tau=t-B}^{t-1} ||z(\tau)||,$$

for some constant A_1 independent of α. We then use a second order expansion of f to conclude that for some A_2, A_3, independent of α, we have

$$f(y(t+1)) \le f(y(t)) - \alpha \sum_{\{i|t\in T^i\}} \frac{\partial f}{\partial x_i}(y(t))\frac{\partial f}{\partial x_i}(x^i(t)) + A_2\alpha^2||z(t)||^2 \le$$

$$\le f(y(t)) - \alpha||z(t)||^2 + \alpha^2 A_1 \sum_{\tau=t-B}^{t-1} ||z(t)||\;||z(\tau)|| + A_2\alpha^2||z(t)||^2 \le$$

$$\le f(y(t)) - \alpha||z(t)||^2 + \alpha^2 A_3 \sum_{\tau=t-B}^{t} ||z(\tau)||^2,$$

which yields

$$f(y(t+1)) \le f(y(0)) - \alpha(1-\alpha(B+1)A_3) \sum_{\tau=0}^{t} ||z(\tau)||^2 . \tag{5.10}$$

If we choose α small enough, so that $1-\alpha(B+1)A_3 > 0$, we can easily show that $z(t)$ and consequently $y(t) - x^i(t)$, converge to zero. If we take any limit point x^* of $\{x^i(t)\}$, it follows that $\frac{\partial f}{\partial x_i}(x^*)=0$. Since $\{x^i(t)\}$ has the same limit points for all i, we conclude that x^* minimizes f.

Q.E.D.

We have included the above proof because it provides the basis for extensions involving overlapping processors and stochastic descent iterations to be discussed later.

Example 4 (continued): A variation of the above proof may be used to prove convergence for that example as well. More interestingly, the discussion at the beginning of this section indicates that the effects of asynchronism are roughly proportional to the cross-derivatives $\frac{\partial^2 f}{\partial x_i \partial x_j}$. Accordingly, convergence may be guaranteed as long as the bound on $t-\tau_k^{ij}(t)$ (which is a measure of asynchronism) is inversely proportional to a bound on the cross-derivatives [35]. Intuitively, this provides a link between communication requirements and the degree of coupling of the divisions of an organization.

Example 5 (continued): An argument similar to the proof of Proposition 5.1 has been used to prove convergence of an asynchronous gradient projection algorithm for the problem of optimal routing in a data network [46].

Let us now return to the proof of Proposition 5.1. It is based on the vector $y(t)$ which serves as a summary of the state of the algorithm at time t and involves effectively a two-time-scale argument: taking α very small, we have $y(t+1) \approx y(t)$ and iteration (5.7b) corresponds to very slow changes. On the other hand, (5.7a) and

(5.8) correspond to much faster variations: within a time period of B units, $x^i(t) - y(t)$ becomes of the order of α, no matter how large the initial difference. This suggests the approximation of each $x^i(t)$ by $y(t)$.

For the above specialization example, a meaningful choice of the aggregate vector $y(t)$ was not hard to guess. We now turn to the case of overlapping processors where such a choice is less evident. For simplicity, we restrict to the case of total overlap.

If we make a slight generalization of Example 6, we obtain the following model:

$$x^i(t+1) = \sum_{\substack{j=1 \\ j\neq i}}^{n} \beta^{ij}(t) z^i_j(t) + \beta^{ii}(t) x^i(t) + \alpha^i(t) z^i_i(t) \tag{5.11}$$

$$z^i_j(t) = x^j(\tau^{ij}_j(t)) . \tag{5.12}$$

For any fixed i,t, the coefficients $\beta^{ij}(t)$ are nonnegative and sum to 1. Moreover if $i\neq j$, $\beta^{ij}(t)=0$, $\forall t\notin T^i_j$, and $\beta^{ij}(t)\geq \bar{\beta}>0$, if $t\in T^i_j$. Finally, $\beta^{ii}(t)\geq \bar{\beta}>0$, $\forall t,i$.

Concerning $z^i_i(t)$, we let

$$z^i_i(t) = \frac{\partial f}{\partial x}(x^i(t)),$$

and

$$\alpha^i(t) = \begin{cases} \alpha^i>0, & t\in T^i \\ 0 , & t\notin T^i , \end{cases}$$

which corresponds again to a distributed gradient algorithm, except that each processor is updating all components of his estimate, thus leading to a certain duplication of computations.

We would like to define an aggregate vector $y(t)$ which summarizes the state of the algorithm at time t. The argument following Proposition 5.1 suggests that $y(t)$ should be chosen so that $y(t) - x^i(t)$ becomes of the order of α fast enough, no matter how large the initial difference is. Fortunately, this is possible. Observing that (5.11),(5.12) are linear evolution equations, there exist scalars $\Phi^{ij}(t|s)$, determined by the coefficients $\beta^{ij}(t)$, such that

$$x^i(t+1) = \sum_{s=1}^{t} \sum_{j=1}^{n} \Phi^{ij}(t|s)\alpha^j(s)z_j^j(s) + \\ + \sum_{j=1}^{n} \Phi^{ij}(t|0)x^j(1) \ . \qquad (5.14)$$

Assume that $t-\tau_j^{ij}(t) \leq B$, $\forall i,j,t$ and that the difference between consecutive elements of T^i is bounded. It can be shown [35] that $\lim_{t\to\infty} \Phi^{ij}(t|s)$ exists, for any i,j,s, is independent of i and will be denoted by $\Phi^j(s)$. Moreover, convergence takes place at the rate of a geometric progression and there exists some $\delta>0$ such that $\Phi^j(s) \geq \delta$, $\forall j,s$. We then define

$$y(t) = \sum_{j=1}^{n} \Phi^j(0)x^j(1) + \sum_{s=1}^{t-1} \sum_{j=1}^{n} \Phi^j(s)\alpha^j(s)z_j^j(s) \ . \qquad (5.15)$$

Comparing with equation (5.14), we can see that this is the common estimate at which all processors would converge if z_j^j was ignored, for all j and for all times larger or equal than t. Equation (5.15) leads to the recursion (compare to (5.9))

$$y(t+1) = y(t) + \sum_{j=1}^{n} \Phi^j(t)\alpha^j(t)z_j^j(t) \qquad (5.16)$$

which is considerably simpler than (5.11),(5.12). Let us observe that $y(t)$ depends on the entire sequence of future coefficients $\beta^{ij}(s)$, $s \geq t$ and is therefore unknown at time t. This is immaterial, however, because y(t), is only an analytical tool used primarily for simplifying the proofs of convergence results. In fact, an argument almost identical to the proof of Proposition 5.1 shows that the same result is valid for this example as well.

We have discussed at considerable length the simplest possible distributed algorithms of a descent type, so as to focus on the main ideas involved. We now mention briefly several directions towards which these results may be generalized. Complete statements and proofs may be found in [35,43].

First, there is nothing particular about the cases of specialization or complete overlap discussed above. Similar results can be proved under the assumption that for each component, a possibly different subset of the set {1,...,n} cooperates in updating that component. To keep the discussion simple, however, we only indicate extensions for the specialization case. The results for the cases of total or partial overlap are very similar.

Proposition 5.1 remain valid if (5.7b) is replaced by a descent assumption of the form

$$x_i^i(t+1) = x_i^i(t) - \alpha\phi_i^i(x^i(t)), \quad t\in T^i,$$

where $\phi_i^i: R^n \to R$ is a continuous function satisfying:

i) $\phi_i^i(x)\,\frac{\partial f(x)}{\partial x_i} \geq 0, \quad \forall x$

ii) $|\phi_i^i(x)| \leq K \left|\frac{\partial f(x)}{\partial x_i}\right|, \quad \forall x$, for some $K \geq 0$

iii) $\sum_i |\phi_i^i(x)| = 0 \Rightarrow x$ minimizes f.

A generalization to a stochastic descent algorithm with multiplicative noise, such as

$$z_i^i(t) = \phi_i^i(x^i(t))(1+w^i(t)) ,$$

with $w^i(t)$ white, is also possible.

More interesting distributed stochastic algorithms are obtained if the noise appears additively rather than multiplicatively, e.g.

$$z_i^i(t) = \phi_i^i(x^i(t)) + w^i(t)$$

where ϕ_i^i satisfies conditions (i)-(iii) above and the noise $w^i(t)$ satisfies for all t

(i) $E[w^i(t)|F_t]=0$ (5.17)

(ii) For some K, $E[|w^i(t)|^2|F_t] \leq K(f(x^i(t))+1)$

where F_t is a σ-field describing the history of the algorithm up to time t. Similarly with centralized algorithms, convergence may be proved only if the step-sizes α^i decrease with time. So, let us assume that $\alpha^i=\alpha^i(t)=1/t^i$, $t \in T^i$, where t^i is the number of times that processor i has received a measurement z_i^i up to time t. Then, the conclusions of Proposition 5.1 hold with probability 1. The proof is based on the supermartingale convergence theorem and resembles the proofs of [40] for related centralized algorithms.

Finally, if f is not convex, one may still prove convergence to a stationary point, exactly as for centralized algorithms.

Example 7: (continued) Several common algorithms for identification (e.g. the Least Mean Squares algorithm, or its normalized version-NLMS) fall into the above framework. Consequently, assuming that the input process u(t) is sufficiently rich and that enough messages are exchanged, it can be shown that certain distributed algorithms will correctly identify the system. A detailed analysis is given in [35].

There is a large class of important stochastic iterative algorithms for which the condition (5.17) fails to hold. Very few global convergence results are available even for centralized such algorithms [34,36] and it is an open question whether some distributed versions of them also converge. However, as in the centralized case [37, 38] one may associate an ordinary differential equation with such an algorithm and prove convergence subject to an assumption that the algorithm returns infinitely often to a bounded region (see [35]). Such results may be used, for example, to demonstrate local convergence of a distributed extended least squares (ELS) algorithm, applied to the ARMAX identification problem in Example 7.

Recall now the time scale separation argument we introduced earlier. Intuitively, for the algorithm to converge it is sufficient that the time scale corresponding to the processor updates- which is in turn determined by the stepsize - is slow when compared to the time scale at which new measurements are obtained. With a decreasing step-size, processor updates become progressively slower. For this reason, convergence may be proved even if the delays $t-\tau_j^{ij}(t)$ increase together with t. In particular, it is sufficient to assume that the delays in equation (2.3) increase with time slower than some polynomial and still obtain convergence. More precisely, we require that there exist $B>0$, $\delta\geq 1$ such that, if $t\geq B(n+1)^{\delta}$, then $\tau_j^{ij}(t)\geq Bn^{\delta}$, $\forall i,j$.

So far in this section we have implicitly assumed that the times that measurements are received are deterministic, although unknown. For stochastic algorithms a new possibility arises: a processor may decide whether to acquire a new measurement based on the current estimate $x^i(t)$, on an old measurement $z^i(t-1)$ or on any other information pertaining to the progress of the algorithm. This results in $\tau^{ij}_j(t)$ being a random variable. For the specialization case convergence may be still proved as long as the condition $t-\tau^{ij}_j(t)\leq B$ holds with probability one. Interestingly enough, this is not always true in the case of overlap and more restrictions are needed to guarantee convergence [35].

6. Convergence of Distributed Processes with Bayesian Updates

In Sections 4 and 5 we considered distributed processes in which a solution is being successively approximated, while the structure of the updates is restricted to be of a special type. In this section we take a different approach and we assume that the estimate computed by any processor at any given time is such that it minimizes the conditional expectation of a cost function, given the information available to him at that time. Moreover, we assume that all processors "know" the structure of the cost function and the underlying statistics, have infinite computing power, and the quality of their estimates is only limited by the availability of posterior information. Whenever a processor receives a measurement z^i_j (possible containing an earlier estimate of another processor) his information changes and a new estimate may be computed.

Formally, let $X= R^m$ be the feasible set, $(\Omega,\mathcal{F},\mathcal{P})$ a probability space and $f: X\times\Omega\to[0,\infty)$ a random cost function which is strongly convex in x for each $\omega\in\Omega$. Let $I^i(t)$ denote the information of processor i at time t, which generates a σ-algebra $\mathcal{F}^i_t \subset \mathcal{F}$. At any time that the information of processor i changes, he updates his estimate according to

$$x^i(t+1) = \arg\min_{x\in X} E[\,f(x,\omega)\,|\,\mathcal{F}^i_t] \ . \qquad (6.1)$$

Assuming that f is jointly measurable, this defines an almost surely unique, $\mathcal{F}^i_t$ - measurable random variable [39].

The information $I^i(t)$ of processor i may change in one of the following ways:

a) New exogenous measurements $z_i^i(t)$ are obtained, so that $I^i(t) = (I^i(t-1), z_i^i(t))$.

b) Measurements $z_j^i(t)$, $j \neq i$, with the value of an earlier estimate of processor j are obtained; that is,

$$z_j^i(t) = x^j(\tau_j^i(t)), \quad 1 \leq \tau_j^i(t) \leq t$$
$$I^i(t) = (I^i(t-1), z_j^i(t)) \tag{6.2}$$

c) Some information in $I^i(t-1)$ may be "forgotten"; that is, $I^i(t) \subset I^i(t-1)$ (or $F_t^i \subset F_{t-1}^i$).

The times at which measurements are obtained as well as the delays are either deterministic or random; if they are random, their statistics are described by (Ω, F, P) and these statistics are known by all processors.

Case 1: Increasing Information. We start by assuming that information is never forgotten, i.e. $F_{t+1}^i \supset F_t^i$, $\forall i,t$. Let $f(x,\omega) = ||x-x^*(\omega)||^2$, where $x^*:\Omega \to R^m$ is an unknown random vector to be estimated. Then,

$$x^i(t+1) = E[x^*(\omega) \, | F_t^i]$$

and by the martingale convergence theorem, $x^i(t)$ converges almost surely to a random variable y^i. Moreover it has been shown that if "enough" measurements of type (6.2) are obtained by each processor, then $y^i = y^j$, $\forall i,j$, almost surely [30,41,44]. If f is not quadratic but strongly convex, the same results are obtained except that convergence holds in the sense of probability and in the $L^2(\Omega,F,\mu)$ sense, where μ is a measure equivalent to P, determined by the function f [39]. However, this scheme is not, strictly speaking, iterative, since $I^i(t)$ increases, and unbounded memory is required.

Case 2: Iterative schemes

The above scheme can be made iterative if we allow processors to forget their past information. For example, let

$$I^i(t) = \begin{cases} \{x^i(t),\ z^i_j(t)\}, & \text{if a measurement } z^i_j(t) \text{ is obtained at time } t \\ \{x^i(t)\}, & \text{otherwise} \end{cases}$$

Let $z^i_j(t) = x^j(\tau^i_j(t))$, $i \neq j$, $\tau^i_j(t) \leq t$. Assuming that "enough" measurements of this type are obtained by each processor, the disagreement $x^i(t)-x^j(t)$ between processors converges to zero as for Case 1 [39]. It has been also shown that $x^i(t+1) - x^i(t)$ converges to zero, for each i, but it is not known whether $x^i(t)$ is guaranteed to convergence or not.

Even though this case corresponds to an iterative algorithm, it may be very hard to implement: The computation of the minimum in (6.1) may be intractable. Also, even if the processors asymptotically converge and agree, there are no guarantees in general about the quality of the final estimate. There is one notable exception where these drawbacks disappear, which we discuss below:

Case 3: Distributed Linear Estimation

Let $f(x,\omega) = ||x-x^*(\omega)||^2$, where x^* is a zero-mean Gaussian scalar random variable to be estimated. Suppose that at time zero each processor obtains measurements

$$z^i_{i,k} = x^* + w^i_k, \quad k=1,\dots,m_i \ , \tag{6.3}$$

where w^i_k are zero-mean Gaussian noises. We allow the noises of different processors to be correlated to each other. Let $I^i(0) = \{z^i_{i,k} | k=1,\dots,m_i\}$. No further measurements of the form (6.3) are obtained after time zero. Subsequently each processor i receives from time to time measurements $z^i_j(t) = x^j(\tau^i_j(t)), \tau^i_j(t) \leq t$, of the other processors' estimates and updates according to

$$x^i(t+1) = E[x^* | I^i(0),\ z^i_j(t)] \ .$$

The timing and delay of these latter measurements is assumed to be deterministic. If we make the assumption that an infinite number of measurements of each type z^i_j is obtained by each processor i, together with an additional assumption that essentially requires that there exists an indirect communication path between every pair of processors then it can be shown that $x^i(t)$ converges in the mean square to the centralized estimate

$$x^* = E[x^*|I^1(0),\dots,I^n(0)],$$

which is the optimal estimate of x* given the total information of all processors [35],[39].

What is interesting about the above algorithm is that it corresponds to a distributed iterative decomposition algorithm for solving the centralized linear estimation problem. The minimization of the cost criterion over a space of dimension $\sum_{i=1}^{n} m_i$, in general, is substituted by a sequence of minimizations along (m_i+1)-dimensional subspaces.

If the noises w_k^i, w_ℓ^i, $i\neq j$, are independent the algorithm converges after finitely many iterations. In general, the algorithm converges linearly but the rate of convergence depends strongly on the number of processors and the angles between certain subspaces of the underlying vector space of random variables (essentially on the correlations between w_k^i and w_ℓ^j, $i\neq j$, see [35],[39]).

REFERENCES

[1] Tannenbaum, A.S., Computer Networks, Prentice Hall, Englewood Cliffs, N.J., 1981.

[2] Schwartz, M., Computer Communication Network Design and Analysis, Prentice Hall, Englewood Cliffs, N.J., 1977.

[3] Kleinrock, L., Queuing Systems, Vol. I & II, J. Wiley, N.Y., 1975.

[4] Schwartz, M., and Stern, T.E., "Routing Techniques Used in Computer Communication Networks," IEEE Trans. on Communications, Vol. COM-28, 1980, pp. 539-552.

[5] Gallager, R.G., "A Minimum Delay Routing Algorithm Using Distributed Computation," IEEE Trans. on Communication, Vol. COM-25, 1977, pp. 73-85.

[6] Bertsekas, D.P., "Optimal Routing and Flow Control Methods for Communication Networks," in Analysis and Optimization of Systems (A. Bensoussan and J.L. Lions, eds.), Springer-Verlag, Berlin and N.Y., 1982, pp. 615-643.

[7] Kung, H.T., "Synchronized and Asynchronous Parallel Algorithms for Multiprocessors," in Algorithms and Complexity, Academic Press, 1976, pp. 153-200.

[8] Bertsekas, D.P., and Gafni, E.M., "Projected Newton Methods and Optimization of Multicommodity Flows," IEEE Trans. on Auto. Control, Vol. AC-28, 1983, pp. 1090-1096.

[9] McQuillan, J.M., Richer, I., and Rosen, E.C., "The New Routing Algorithm for the ARPANET," IEEE Trans. on Communications, Vol. COM-28, 1980, pp. 711-719.

[10] Chazan, D., and Miranker, W., "Chaotic Relaxation," Linear Algebra and Applications, Vol. 2, 1969, pp. 199-222.

[11] Baudet, G.M., "Asynchronous Iterative Methods for Multiprocessors," Journal of the ACM, Vol. 2, 1978, pp. 226-244.

[12] Bertsekas, D.P., "Distributed Dynamic Programming," IEEE Trans. on Aut. Control, Vol. AC-27, 1982, pp. 610-616.

[13] Bertsekas, D.P., "Distributed Asynchronous Computation of Fixed Points," Math. Programming, Vol. 27, 1983, pp. 107-120.

[14] McQuillan, J., Falk, G., and Richer, I., "A Review of the Development and Performance of the ARPANET Routing Algorithm," IEEE Trans. on Communications, Vol. COM-26, 1978, pp. 1802-1811.

[15] Zangwill, W.I., Nonlinear Programming, Prentice Hall, Englewood Cliffs, N.Y., 1969.

[16] Luenberger, D.G., Introduction to Linear and Nonlinear Programming, Addison-Wesley, Reading, MA, 1973.

[17] Ortega, J.M. and Rheinboldt, W.C., Iterative Solution of Nonlinear Equations in Several Variables, Academic Press, N.Y., 1970.

[18] Polak, E., Computational Methods in Optimization: A Unified Approach, Academic Press, N.Y., 1971.

[19] Poljak, B.T., "Convergence and Convergence Rate of Iterative Stochastic Algorithms," Automation and Remote Control, Vol. 12, 1982, pp. 83-94.

[20] Bertsekas, D.P., Dynamic Programming and Stochastic Control, Academic Press, N.Y., 1976.

[21] Sandell, N.R., Jr., P. Varaiya, M. Athans and M. Safonov, "Survey of Decentralized Control Methods for Large Scale Systems," IEEE Trans. on Aut. Control, Vol. AC-23, No. 2, 1978, pp. 108-128.

[22] Ho, Y.C., "Team Decision Theory and Information Structure," IEEE Proceedings, Vol. 68, No. 6, 1980, pp. 644-654.

[23] Yao, A.C., "Some Complexity Questions Related to Distributed Computing," Proc. of the 11th STOC, 1979, pp. 209-213.

[24] Papadimitriou, C.H. and M. Sipser, "Communication Complexity," Proc. of the 14th STOC, 1982, pp. 196-200.

[25] Witsenhausen, H.S., "A Counterexample in Stochastic Optimum Control," SIAM J. Control, Vol. 6, No. 1, 1968, pp. 138-147.

[26] Papadimitriou, C.H. and J.N. Tsitsiklis, "On the Complexity of Designing Distributed Protocols," Information and Control, Vol. 53, 1982, pp. 211-218.

[27] Tenney, R.R. and N.R. Sandell, Jr., "Detection with Distributed Sensors," IEEE Trans. on Aerospace and Electronic Systems, Vol. AES-17, No. 4, 1981, pp. 501-509.

[28] Willsky, A.S., M. Bello, D.A. Castanon, B.C. Levy and G. Verghese, "Combining and Updating of Local Estimates and Regional Maps along Sets of One-Dimensional Tracks," IEEE Trans. on Aut. Control, Vol. AC-27, No. 4, 1982, pp. 799-813.

[29] Sanders, C.W., E.C. Tacker, T.D. Linton, R.Y.-S. Ling, "Specific Structures for Large Scale State Estimation Algorithms Having Information Exchange," IEEE Trans. on Aut. Control, Vol. AC-23, No. 2, 1978, pp. 255-261.

[30] Borkar, V. and P. Varaiya, "Asymptotic Agreement in Distributed Estimation," IEEE Trans. on Aut. Control, Vol. AC-27, No. 3, 1982, pp. 650-655.

[31] Arrow, K.J. and L. Hurwicz, "Decentralization and Computation in Resource Allocation," in Essays in Economics and Econometrics, R.W. Pfouts, Ed., Univ. of North Carolina Press, Chapel Hill, NC, 1960, pp. 34-104.

[32] Meerkov, S.M., "Mathematical Theory of Behavior-Individual and Collective Behavior of Retardable Elements," Mathematical Biosciences, Vol. 43, 1979, pp. 41-106.

[33] Astrom, K.J. and P. Eykhoff, "System Identification-A Survey," Automatica, Vol. 7, 1971, pp. 123-162.

[34] Solo, V., "The Convergence of AML," IEEE Trans. on Aut. Control, Vol. AC-24, 1979, pp. 958-963.

[35] Tsitsiklis, J.N., "Problems in Decentralized Decision Making and Computation," Ph.D. Thesis, Dept. of Electrical Engineering and Computer Science, MIT, Cambridge, MA, in preparation.

[36] Nemirovsky, A.S., "On a Procedure for Stochastic Approximation in the Case of Dependent Noise," Engineering Cybernetics, 1981, pp. 1-13.

[37] Ljung, L., "Analysis of Recursive Stochastic Algorithms," IEEE Trans. on Auto. Control, Vol. AC-22, No. 4, 1977, pp. 551-575.

[38] Ljung, L., "On Positive Real Transfer Functions and the Convergence of Some Recursive Schemes," IEEE Trans. on Auto. Control, Vol. AC-22, No. 4, 1977, pp. 539-551.

[39] Tsitsiklis, J.N. and M. Athans, "Convergence and Asymptotic Agreement in Distributed Decision Problems," IEEE Trans. on Aut. Control, Vol. AC-29, 1984, pp. 42-50.

[40] Poljak, B.T. and Y.Z. Tsypkin, "Pseudogradient Adaptation and Training Algorithms," Automation and Remote Control, No. 3, 1973, pp. 45-68.

[41] Geanakoplos, J.D. and H.M. Polemarchakis, "We Can't Disagree Forever," Institute for Mathematical Studies in the Social Sciences, Technical Report No. 277, Stanford University, Stanford, CA, 1978.

[42] Awerbuch, B., "An Efficient Network Synchronization Protocol," Technion-Electrical Engineering Pub. No. 458, Haifa, Israel, Nov. 1983.

[43] Tsitsiklis, J.N., Bertsekas, D.P., and M. Athans, "Distributed Asynchronous Deterministic and Stochastic Gradient Optimization Algorithms," IEEE Trans. on Auto. Control, (submitted).

[44] Washburn, R.B., and D. Teneketzis, "Asymptotic Agreement Among Communicating Decision Makers," Proc. of 1983 Automatic Control Conference, San Francisco, CA. pp. 1331-1336.

[45] Mosely, J., "Asynchronous Distributed Flow Control Algorithms," Ph.D. Thesis, Dept. of Electrical Engineering, M.I.T., June 1984.

[46] Tsitsiklis, J.N. and D.P. Bertsekas, "Distributed Asynchronous Optimal Routing for Data Networks," Proceedings of the 23d CDC, Las Vegas, Nevada, December 1984.