January 1984
revised November 1984 LIDS-P-1425

DISTRIBUTED ASYNCHRONOUS DETERMINISTIC AND STOCHASTIC GRADIENT OPTIMIZATION ALGORITHMS[†]

by

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

ABSTRACT

We present a model for asynchronous distributed computation and then proceed to analyze the convergence of natural asynchronous distributed versions of a large class of deterministic and stochastic gradient-like algorithms. We show that such algorithms retain the desirable convergence properties of their centralized counterparts, provided that the time between consecutive communications between processors and communication delays are not too large.

[†]Research supported by Grants ONR/N00014-77-C-0532 and NSF-ECS-8217668.

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

1. INTRODUCTION

Many deterministic and stochastic iterative algorithms admit a natural distributed implementation [1,4,5] whereby several processors perform computations and exchange messages with the end-goal of minimizing a certain cost function. If all processors communicate to each other their partial results at each instance of time and perform computations synchronously, the distributed algorithm is mathematically equivalent to a single processor (serial) algorithm and its convergence may be studied by conventional means. Synchronous algorithms may have, however, certain drawbacks: a) Synchronism may be hard to enforce, or its enforcement may introduce substantial overhead. b) Communication delays may introduce bottlenecks to the speed of the algorithm (the time required for one stage of the algorithm will be constrained by the slowest communication channel). c) Synchronous algorithms may require far more communications than are actually necessary. d) Even if all processors are equally powerful some will perform certain computations faster than others, due solely to the fact that they operate on different inputs. This in turn, may lead to having many processors idle for a large proportion of time. For these reasons, we choose to study asynchronous distributed iterative optimization algorithms in which each processor does not need to communicate to each other processor at each time instance; also, processors may keep performing computations without having to wait until they receive the messages that have been transmitted to them; processors are allowed to remain idle some of the time; finally, some processors may perform computations faster than others. Such algorithms can alleviate communication overloads and they are not excessively slowed down by neither communication delays, nor by differences in the time it takes processors to perform one computation. (A similar discussion of the merits of asynchronous algorithms is provided by H.T. Kung [10].)

The algorithms that we consider are gradient-like, meaning that the (expected) updates by each processor are in a descent direction with respect to a cost function being minimized. There may be certain limitations to our approach, because we restrict attention to a special class of algorithms, whereas it could be the case that algorithms with a different structure may offer some advantages over the ones we propose. However, the problem of finding an "optimal" distributed algorithm, which minimizes, say, the amount of information exchanged between processors, or some other measure of communication and computation, can be very hard or intractable [15,20] (NP-complete or worse). For this reason, we prefer to assume a fixed structure and then proceed to investigate the amount of asynchronism and the magnitude of the communication delays that may be tolerated without adversely affecting the convergence of the algorithm.

In Section 2 we present the model of distributed computation to be employed. In this model, there is a number of processors who perform certain computations and update some of the components of a vector stored in their memory. In the meanwhile, they exchange messages, thus informing each other about the results of their latest computations. Processors who receive messages use them either to update directly some of the components of the vector in their memory, or they may combine the message with the outcome of their own computations, by forming a convex combination. Very weak assumptions are made about the relative timing and frequency of computations or message transmissions by the processors.

In Section 3 we employ this model of computation and also assume that the (possibly random) updates of each processor are expected to be in a descent direction, when conditioned on the past history of the algorithm. Our main results show that, under certain assumptions, asynchronous distributed algorithms have similar convergence properties as their centralized counterparts, provided that the time between consecutive communications between processors plus communication delays are not too large. We distinguish two cases: a) constant step-size algorithms (e.g. deterministic gradient-type algorithms) in which the time between consecutive communications has to be bounded for convergence to be guaranteed and, b) decreasing step-size algorithms (e.g. stochastic approximation-type algorithms) for which convergence is proved even if the time between consecutive communications increases without bounds as the algorithm proceeds. Section 2 and 3 are developed in parallel with a variety of examples which are used to motivate and explain the formal assumptions that are being introduced.

Finally, Section 4 suggests some extensions and areas of application, together with our conclusions. Appendix A contains the proof of Lemma 2.1. Appendix B contains the proofs of our main results.

2. A MODEL OF DISTRIBUTED COMPUTATION

We present here the model of distributed computation employed in this paper. We also define the notation and conventions to be followed. Related models of distributed computation have been used in [3,4,5,8], in which each processor specialized in updating a different component of some vector. The model developed here is more general, in that it allows different processors to update the same component of some vector. If their individual updates are different, their disagreement is (asymptotically) eliminated through a process of communicating and combining their individual updates. In such a case, we will say that there is overlap between processors. Overlapping processors are probably not very useful in the context of deterministic algorithms, unless redundancy is intended to provide a certain degree of fault tolerance and/or safeguarding against malfunction of a particular processor. For stochastic algorithms, however, overlap essentially amounts to having different processors obtain noisy measurements of the same unknown quantity and effectively increases the "signal-to-noise ratio."

Let $H_1, H_2, \ldots, H_L$ be finite dimensional real vector spaces[1] and let $H=H_1 x H_2 x \ldots x H_L$, which we endow with the Euclidean norm. If $x=(x_1,x_2,\ldots,x_L)$, $x_\ell \in H_\ell$, we will refer to x_ℓ as the ℓ-th component of x.

Let $\{1,\ldots,M\}$ be the set of processors that participate in the distributed computation. As a general rule concerning notation, we use subscripts to indicate a component of an element of H, superscripts to indicate an associated processor; we indicate time by an argument that follows.

The algorithms to be considered evolve in discrete time. Even if a distributed algorithm is asynchronous and communication delays are real (i.e., not integer) variables, the events of interest (an update by some processor, transmission or reception of a message) may be indexed by a discrete variable; so, the restriction to discrete time entails no loss of generality.

It is important here to draw a distinction between "global" and "local" time. The time variable we have just referred to corresponds to a global clock. Such a global clock is needed only for analysis purposes: it is the clock of an analyst who watches the operation of the system. On the other hand the processors may be working without having access to a global clock. They may have access to a local clock or to no clock at all. We will see later that our results, based on the existence of a

[1] All of our results generalize to the case where each H_ℓ is a Banach space. No modifications whatsoever are needed in the assumptions or the proofs except that matrices should be now called linear operators and that expressions like $\nabla J(x)$ should be interpreted as elements of the dual of H rather than vectors in H.

global clock, may be used in a straightforward way to prove convergence of algorithms implemented on the basis of local clocks only.

We assume that each processor has a buffer in its memory in which it keeps some element of H. The value stored by the i-th processor at time n is denoted by $x^i(n)$. At time n, each processor may receive some exogenous measurements and/or perform some computations. This allows it to compute a "step" $s^i(n) \in H$, to be used in evaluating the new vector $x^i(n+1)$. Besides their own measurements and computations, processors may also receive messages from other processors, which will be taken into account in evaluating their next vector. The process of communications is assumed to be as follows:

At any time n, processor i may transmit some (possibly all) of the components of $x^i(n)$ to some (possibly all or none) of the other processors. (In a physical implementation, messages do not need to go directly from their origin to their destination; they may go through some intermediate nodes. Of course, this does not change the mathematical model presented here). We will assume that communication delays are bounded.[2] For convenience, we also assume that for any pair (i,j) of processors, for any component x_ℓ and any time n, processor i may receive at most one message originating from processor j and containing an element of H_ℓ. This leads to no significant loss of generality: for example, a processor that receives two messages simultaneously could keep only the one which was most recently sent; if messages do not carry timestamps, there could be some other arbitration mechanism. Physically, of course, simultaneous receptions are impossible; so, a processor may always identify and keep the most recently received message, even if all messages arrived at the same discrete time n.

If a message from processor j, containing an element of H_ℓ is received by processor i ($i \neq j$) at time n, let $t_\ell^{ij}(n)$ denote the time that this message was sent.

[2] For algorithms with decreasing-step-size this assumption may be relaxed.

Therefore, the content of such a message is precisely $x_\ell^j(t_\ell^{ij}(n))$. Naturally, we assume that $t_\ell^{ij}(n)\leq n$. For notational convenience, we also let $t_\ell^{ii}(n)=n$, for all i,ℓ,n. We will be assuming that the algorithm starts at time 1; accordingly, we assume that that $t_\ell^{ij}(n)\geq 1$. Finally, we denote by T_ℓ^{ij} the set of all times that processor i receives a message from processor j, containing an element of H_ℓ. To simplify matters we will assume that, for any i,j,ℓ, the set T_ℓ^{ij} is either empty or infinite.

Once processor i has received the messages arriving at time n and has also evaluated $s^i(n)$, it evaluates its new vector $x^i(n+1)\in H$ by forming (componentwise) a convex combination of its old vector and the values in the messages it has just received, as follows:

$$x_\ell^i(n+1) = \sum_{j=1}^{M} a_\ell^{ij}(n)x_\ell^j(t_\ell^{ij}(n)) + \gamma^i(n)s_\ell^i(n), \quad n\geq 1, \tag{2.1}$$

where $s_\ell^i(n)$ is the ℓ-th component of $s^i(n)$ and the coefficients $a_\ell^{ij}(n)$ are scalars satisfying:

$$\text{(i)} \qquad a_\ell^{ij}(n)\geq 0, \qquad \forall i,j,\ell,n, \tag{2.2}$$

$$\text{(ii)} \qquad \sum_{j=1}^{M} a_\ell^{ij}(n)=1, \qquad \forall i,\ell,n, \tag{2.3}$$

$$\text{(iii)} \qquad a_\ell^{ij}(n)=0, \qquad \forall n\notin T_\ell^{ij},\ i\neq j. \tag{2.4}$$

Remarks:

1. Note that $t_\ell^{ij}(n)$ has been defined only for those times n that processor i receives a message of a particular type, i.e. for $n\in T_\ell^{ij}$. However, whenever $t_\ell^{ij}(n)$ is undefined, we have assumed above that $a_\ell^{ij}(n)=0$, so that equation (2.1) has an unambiguous meaning.

2. When we refer to a processor performing a "computation," we mean the evaluation and addition of the term $\gamma^i(n)s^i_\ell(n)$ in (2.1). With this terminology, forming the convex combination in (2.1) is not called a computation. We denote by T^i_ℓ the set of all times that processor i performs a computation involving the ℓ-th component. Whenever $n\notin T^i_\ell$, it is understood that $s^i_\ell(n)$ in (2.1) equals zero. We assume again that for any i,ℓ the set T^i_ℓ is either infinite or empty. Accordingly, processor i will be called computing, or non-computing, for component ℓ.

3. The quantities $\gamma^i(n)$ in (2.1) are nonnegative scalar step-sizes. These step-sizes may be constant (e.g. $\gamma^i(n)=\gamma_o, \forall n$), or time-varying, e.g. $\gamma^i(n)=1/t^i_n$, where t^i_n is the number of times that processor i has performed a computation up to time n. Notice that with such a choice each processor may evaluate its step-size using only a local counter rather than a global clock.

4. The envisaged sequence of events underlying (2.1) at any time n, is as follows: For each component ℓ processor i

   (i) Transmits $x^i_\ell(n)$ to processors and receives messages $x^j_\ell(t^{ij}_\ell(n))$ from processors j for which $n\in T^{ij}_\ell$.

   (ii) Computes $s^i_\ell(n)$ if $n\in T^i_\ell$ and sets $s^i_\ell(n)=0$ otherwise.

   (iii) Evaluates $x^i_\ell(n+1)$ according to (2.1).

Examples

We now introduce a collection of simple examples representing various classes of algorithms we are interested in, so as to illustrate the nature of the assumptions to be introduced later. We actually start with a broad classification and then proceed to more special cases. In these examples, we model the message receptions and transmissions (i.e. the sets T_ℓ^{ij} and the variables $t_\ell^{ij}(n)$), the times at which computations are performed (i.e. the sets T_ℓ^{i}) and the combining coefficients $a_\ell^{ij}(n)$ as deterministic. (This does not mean, however, that they have to be a priori known by the processors).

Specialization: This is the case considered by Bertsekas [4,5], where each processor updates a particular component of the x-vector specifically assigned to it and relies on messages from the other processors for the remaining components. Formally:

(i) $M=L$. (There are as many processors as there are components).

(ii) $s_\ell^i(n)=0$, $\forall \ell \neq i$, $\forall n$. (A processor may update only its own component; $T_\ell^i=\phi$, $\forall i \neq \ell$).

(iii) Processor j only sends messages containing elements of H_j; if processor i receives such a message, it uses it to update x_j^i by setting x_j^i equal to the value received. Equivalently,

(a) If $i \neq j$ and $j \neq \ell$, then $T_\ell^{ij}=\phi$ and $a_\ell^{ij}(n)=0$, $\forall n$.

(b) If processor i receives a message from processor j at time n, i.e. if $n \in T_j^{ij}$, then $a_j^{ij}(n)=1$. Otherwise, $a_j^{ij}(n)=0$, and $a_j^{ii}(n)=1$.

Overlap: This is the other extreme, at which $L=1$ (we do not distinguish components of elements of H), messages contain elements of H (not just components) and each processor may update any component of x. (For this case subscripts are redundant and will be omitted.)

We now let H be finite-dimensional and assume that $J: H \to [0,\infty)$ is a continuously differentiable nonnegative cost function with a Lipschitz continuous derivative.

Example I: Deterministic Gradient Algorithm; Specialization. Let $\gamma^i(n) = \gamma_o > 0, \forall n,i$. At each time $n \in T_i^i$ that processor i updates x_i^i, it computes $s_i^i(n) = -\frac{\partial J}{\partial x_i}(x^i(n))$ and lets $s_j^i(n)=0$, for $j \neq i$. We assume that each processor i communicates its component x_i^i to every other processor at least once every B_1 time units, for some constant B_1. Other than this restriction, we allow the transmission and reception times to be arbitrary. (A related stochastic algorithm could be obtained by letting $s_i^i(n) = -\frac{\partial J}{\partial x_i}(x^i(n))(1+w_i^i(n))$, where $w_i^i(n)$ is unit variance white noise, independent for different i's).

Example II: Newton's Method; Overlap. For simplicity we assume that there are only two processors (M=2). Let $\gamma^i(n) = \gamma_o > 0, \forall n$. We also assume that J is twice continuously differentiable, strictly convex and its Hessian matrix, denoted by G(x), satisfies $0 < \delta_1 I \leq G(x) \leq \delta_2 I$, $\forall x \in H$. At each time $n \in T^i$, processor i computes $s^i(n) = -G^{-1}(x^i(n)) \frac{\partial J}{\partial x}(x^i(n))$. For $n \notin T^i$, $s^i(n)=0$. If at time n processor 1 (respectively, 2) receives a message $x^2(t^{12}(n))$(resp.. $x^1(t^{21}(n))$), it updates its vector by $x^1(n+1) = a_{11}x^1(n) + a_{12}x^2(t^{12}(n)) + \gamma^1(n)s^1(n)$, (resp. $x^2(n+1) = a_{21}x^1(t^{21}(n)) + a_{22}x^2(n) + \gamma^2(n)s^2(n)$). Here we assume that $0 < a_{ij} < 1$ and that $a_{11}+a_{12}=a_{21}+a_{22}=1$. For other times n the same formula is used with $a_{12}=0$ ($a_{21}=0$). We make the same assumptions on transmission and reception times as in Example 1.

Example III: Distributed Stochastic Approximation; Specialization. Let $\gamma^i(n)$ be such that, for some positive constants A_1, A_2, $A_1/n \leq \gamma^i(n) \leq A_2/n$, $\forall n$. Notice that the implementation of such a stepsize only requires a local clock that runs in the same time scale (i.e. within a constant factor) as the global clock. For $n \in T^i$, let $s_i^i(n) = -\frac{\partial J}{\partial x_i}(x^i(n)) + w_i^i(n)$. Also, $s_j^i(n)=0$, for $i \neq j$ and for all n.

We assume that $w_i^i(n)$, conditioned on the past history of the algorithm has zero mean and that $E[||w_i^i(n)||^2 \; |x^i(n)] \leq K(||\nabla J(x^i(n))||^2+1)$, for some constant K. We assume that for some $B_1 \geq 0$, $\beta \geq 1$ and for all n, each processor communicates its component x_i^i to every other processor at least once during the time interval $[B_1 n^\beta, B_1(n+1)^\beta]$. Other than the above restriction, we allow transmission and reception times to be arbitrary. Notice that the above assumptions allow the time between consecutive communications to grow without bound.

Example IV: Distributed Stochastic Approximation; Overlap. Let $\gamma^i(n)$ be as in Example III and let M=2. For $n \in T^i$, let $s^i(n) = -\frac{\partial J}{\partial x}(x^i(n)) + w^i(n)$, where $w^i(n)$ is as in Example III. We make the same assumptions on transmission and reception times as in Example III. Whenever a message is received, a processor combines its vector with the content of that message using the combining rules of Example II.

Example V: This example is rather academic but will serve to illustrate some of the ideas to be introduced later. Consider the case of overlap, assume that H is one-dimensional, and let $\gamma^i(n)=1$, $\forall n$. Assume that, at each time n, either all processors communicate to each other, or no processor sends any message. Let the communication delays be zero (so, $t^{ij}(n)=n$, whenever $t^{ij}(n)$ is defined) and assume that $a^{ij}(n) = a^{ij}$ (constant) at thosetimes n that messages are exchanged. We define vectors $x(n) = (x^1(n),\ldots,x^M(n))$ and $s(n) = (s^1(n),\ldots,s^M(n))$. Then, the algorithm (2.1) may be written as

$$x(n+1) = A(n)x(n) + s(n) \tag{2.5}$$

For each time n, either A(n)=I (no communications) or A(n)=A, the matrix consisting of the coefficients a^{ij}. The latter is a "stochastic" matrix: it has nonnegative entries and each row sums to 1. We assume that a^{ij} is positive. It follows that $\overline{A} = \lim_{n\to\infty} A^n$ exists and has identical rows with positive elements. We assume that the time between consecutive communications is bounded but otherwise arbitrary. Clearly then, $\lim_{n\to\infty} \prod_{m=k}^{n} A(m)=\overline{A}$, for all k. This example corresponds to a set of processors who individually solve the same problem and, from time to time, simultaneously exchange their partial results. It is interesting to compare equation (2.5) with the generic equation

$$x(n+1) = x(n) + \gamma(n)s(n)$$

which arises in centralized algorithms.

Assumptions on the communications and the combining coefficients

We now consider a set of assumptions on the nature of the communications and combining process, so that the preceding examples appear as special cases. In formulating an appropriate set of assumptions, there is a trade-off between generality and ease of verification. The assumptions below are not the most general possible, but are very easy to enforce. Some generalizations will be suggested later.

For each component $\ell\in\{1,\dots,L\}$ we introduce a directed graph $G_\ell=(V,E_\ell)$ with nodes $V=\{1,\dots,M\}$ corresponding to the set of processors. An edge (j,i) belongs to E_ℓ if and only if T_ℓ^{ij} is infinite, that is, if and only if processor j sends (in the long run) an infinite number of messages to processor i with a value of the ℓ-th component x_ℓ^j.

Assumption 2.1: For each component $\ell\in\{1,\dots,L\}$, the following hold:

a) There is at least one computing processor for component ℓ.

b) There is a directed path in G_ℓ, from every computing processor (for component ℓ) to every other processor (computing or not).

c) There is some $\alpha>0$ such that:

(i) If processor i receives a message from processor j at time n (i.e. if $n\in T_\ell^{ij}$), then $a_\ell^{ij}(n)\geq\alpha$.

(ii) For every computing processor i, $a_\ell^{ii}(n)\geq\alpha$, $\forall n$.

(iii) If processor i has in-degree[3] (in G_ℓ) larger or equal than 2, then $a_\ell^{ii}(n)\geq\alpha$, $\forall n$.

Let us pause to indicate the intuitive content of part (c) of Assumption 2.1. Part (i) states that a processor should not ignore the messages it receives. Part (ii) requires the past updates of any computing processor to have a lasting effect on its state of computation. Finally, part (iii) implies that if processor i receives messages from two processor (say i_1,i_2), it does not forget the effects of messages of processor i_1 upon reception of a message from processor i_2. These conditions, together with part (b) of the Assumption, guarantee that any update by any computing processor has a lasting effect on the states of computation of all other processors.

Assumption 2.2: The time between consecutive transmissions of component x_ℓ^j from processor j to processor i is bounded by some $B_1\geq0$, for all $(j,i)\in E_\ell$.

Assumption 2.3: There are constants $B_1\geq0$, $\beta\geq1$ such that, for any $(j,i)\in E_\ell$, and for any n, at least one message x_ℓ^j is sent from processor j to processor i during the time interval $[B_1n^\beta, B_1(n+1)^\beta]$. Moreover, the total number of messages transmitted and/or received during any such interval is bounded.

Assumption 2.4: Communication delays are bounded by some $B_0\geq0$, i.e. for all i,j,ℓ and $n\in T_\ell^{ij}$ we have $n-t_\ell^{ij}(n)\leq B_0$.

[3] The in-degree of a processor (node) i (in G_ℓ) is the number of edges in E_ℓ pointing to node i.

Note that Assumption 2.2 is a special case of 2.3, with $\beta=1$. Assumption 2.1 holds for all the examples introduced above. Assumption 2.2 holds for Examples I, II,V; Assumption 2.3 holds for Examples III,IV, except for its last part which has to be explicitly introduced.

Equation (2.1) which defines the structure of the algorithm is a linear system driven by the steps $s_j^i(n)$. In the special case where communication delays are zero, we have $t_\ell^{ij}(n)=n$, and (2.1) becomes a linear system with state vector $(x^1(n),\ldots,x^M(n))$. Equation (2.5) of Example V best illustrates this situation. In general, however, the presence of communication delays necessitates an augmented state if a state space representation is desired[4]. (Notice that actually (2.1) defines a decoupled set of linear systems, one for each component $\ell \in \{1,\ldots,L\}$.) Exploiting linearity, we conclude that there exist scalars $\Phi^{ij}(n|k)$, for $n\geq k$, such that

$$x_\ell^i(n) = \sum_{j=1}^{M} \Phi_\ell^{ij}(n|0)x_\ell^j(1) + \sum_{k=1}^{n-1} \sum_{j=1}^{M} \gamma^j(k)\Phi_\ell^{ij}(n|k)s_\ell^j(k). \tag{2.6}$$

The coefficients $\Phi_\ell^{ij}(n|k)$ are determined by the sequence of transmission and reception times and the combining coefficients. Consequently, they are unknown, in general. Nevertheless, they have the following qualitative properties.

<u>Lemma 2.1</u>: (i) $$0\leq\Phi_\ell^{ij}(n|k), \quad \forall i,j,\ell,n\geq k, \tag{2.7}$$

$$\sum_{j=1}^{M} \Phi_\ell^{ij}(n|k)\leq 1, \quad \forall i,\ell,n\geq k . \tag{2.8}$$

(ii) Under Assumptions 2.1, 2.4 and either Assumption 2.2 or 2.3, $\lim_{n\to\infty} \Phi_\ell^{ij}(n|k)$ exists, for any i,j,k,ℓ. The limit is independent of i and will be denoted by $\Phi_\ell^j(k)$.

[4] Such an augmented state should incorporate all messages that have been transmitted but not yet received. Since we are assuming bounded communication delays, there can only be a bounded number of such messages and the augmented system may be chosen finite dimensional.

Moreover, there is a constant $\eta>0$ such that, if j is a computing processor for component ℓ, then

$$\Phi_\ell^j(k)\geq\eta, \quad \forall k. \tag{2.9}$$

The constant η, depends only on the constants introduced in our assumptions (i.e. B_o, B_1, β, α).

(iii) Under Assumptions 2.1, 2.2, 2.4, there exist $d\in[0,1)$, $B\geq 0$ (depending only on B_o, B_1, α) such that

$$\max_{i,j} |\Phi_\ell^{ij}(n|k) - \Phi_\ell^j(k)|\leq Bd^{n-k}, \quad \forall \ell, n\geq k . \tag{2.10}$$

(iv) Under Assumptions 2.1, 2.3, 2.4, there exist $d\in[0,1)$, $\delta\in(0,1]$, $B\geq 0$ (depending only on B_o, B_1, β, α) such that

$$\max_{i,j} |\Phi_\ell^{ij}(n|k) - \Phi_\ell^j(k)|\leq Bd^{n^\delta-k^\delta}, \quad \forall \ell, n\geq k . \tag{2.11}$$

Proof: See Appendix A.

In the light of equation (2.6), Lemma 2.1 admits the following interpretation: part (ii) states that if all processors cease updating (that is if they set $s^i(n)=0$) from some time on, they will asymptotically converge to a common limit. Moreover, this common limit depends by a non-negligible factor on all past updates of all computing processors. Parts (iii) and (iv) quantify the natural relationship between the frequency of interprocessor communications and the speed at which agreement is reached.

For any pair (i,j) of processors we may also define a linear transformation $\Phi^{ij}(n|k):H\to H$ by

$$\Phi^{ij}(n|k) = (\Phi_1^{ij}(n|k)x_1,\ldots,\Phi_L^{ij}(n|k)x_L), \tag{2.12}$$

where $x=(x_1,\ldots,x_L)$. Note that if each H_ℓ is one-dimensional then $\Phi^{ij}(n|k)$ may be represented by a diagonal matrix. In general, it corresponds to a block-diagonal matrix, each block being a constant multiple of the identity matrix of suitable dimension. Clearly, $\lim_{n\to\infty}\Phi^{ij}(n|k)$ also exists, is independent of i and will be denoted by $\Phi^j(k)$.

We can now define a vector $y(n)\in H$ by

$$y(n) = \sum_{j=1}^{M} \Phi^j(0)x^j(1) + \sum_{k=1}^{n-1}\sum_{j=1}^{M} \gamma^j(k)\Phi^j(k)s^j(k) \tag{2.13}$$

and note that y(n) is recursively generated by

$$y(n+1) = y(n) + \sum_{j=1}^{M} \gamma^j(n)\Phi^j(n)s^j(n) \ . \tag{2.14}$$

The vector y(n) is the element of H at which all processors would asymptotically agree if they were to stop computing (but keep communicating and combining) at a time n. It may be viewed as a concise global summary of the state of computation at time n, in contrast with the vectors $x^i(n)$ which are the local states of computation; it allows us to look at the algorithm from two different levels: an aggregate and a more detailed one. We will see later that this vector y(n) is also a very convenient tool for proving convergence results. We have noted earlier that equation (2.1) is a linear system. However, it is a fairly complicated one, whereas the recursion (2.14) corresponds to a very simple linear system in standard state space form. The content of the vector y(n) and of the $\Phi^j(n)$'s is easiest to visualize in two special cases:

Specialization: (e.g. Examples I and III). Here y(n) takes each component from the processor who specializes in that component. That is, $y(n) = (x_1^1(n),\ldots,x_M^M(n))$. Accordingly, $\Phi_j^i(n)=0$, for $i\neq j$, and $\Phi_j^j(n)=1$.

Example V: Here $\Phi^{ij}(n|k)$ is the ij-th entry of the matrix $\prod_{m=k+1}^{n-1} A(m)$. It follows that the limit of $\Phi^{ij}(n|k)$ is the ij-th entry of $\bar{A}$, which by our assumptions depends only on j. Moreover, y(n) equals any component of $\bar{A}x(n)$. (All components are equal by our assumptions). If we multiply both sides of (2.5) by $\bar{A}$ and note that $\bar{A}A(n)=\bar{A}$, we obtain $y(n+1)=y(n) + \sum_{j=1}^{M} \bar{A}_{ij} s^j(n)$, which is precisely (2.14).

The model of computation introduced in this section may be generalized in several directions [7,19]. To name a few examples, the updating rules of each processor need not have the linear structure of equation (2.1); also, it may be convenient to communicate other information (e.g. derivatives of the cost function) and not just the values of components of x. However, the present model is sufficient for the class of algorithms under consideration. Let us also mention that, while all of the above examples refer to either specialization or overlap, we may think of intermediate situations in which some of the components are updated by a single processor while some of the components are updated by all processors simultaneously (partial overlap).

Finally, Assumptions 2.1, 2.4 are unnecessary, as long as the conclusions of Lemma 2.1 may be somehow independently verified.

3. CONVERGENCE RESULTS

There is a large number of well-known centralized deterministic and stochastic optimization algorithms which have been analyzed using a variety of analytical tools [2,11,12,16]. A large class of them, the so-called "pseudo-gradient" algorithms [16], have the distinguishing feature that the (expected) direction of update (conditioned upon the past history of the algorithm) is a descent direction with respect to the cost function to be minimized. The Examples of Section 2 certainly have such a property. Reference [16] presents a larger list of examples. It is also shown there that the development of results for pseudo-gradient algorithms leads easily to results for broader classes of algorithms, such as Kiefer-Wolfowitz stochastic approximation. In this section we present convergence results for the natural distributed asynchronous versions of pseudo-gradient algorithms. We adopt the model of computation and the corresponding notation of Section 2.

We allow the initialization $\{x^1(1),\ldots,x^M(1)\}$ of the algorithm to be random, with finite mean and variance. We also allow the updates $s^i(n)$ of each processor to be random. On the other hand, we assume that $\gamma^i(n)$ is deterministic; we also model the combining coefficients, $a_\ell^{ij}(n)$ and the sequence of transmission and reception times as being deterministic. This is not a serious restriction because they do not need to be known by the processors in advance, in order to carry out the algorithm. We assume that all random variables of interest are defined on a probability space (Ω,F,P). We introduce $\{F_n\}$, an increasing sequence of σ-fields contained in F and describing the history of the algorithm up to time n. In particular, F_n is defined as the smallest σ-field such that $s^i(k)$, $k\leq n-1$, and $x^i(1)$, $i\in\{1,\ldots,M\}$ are F_n-measurable.

We assume that the objective of the algorithm is to minimize a nonnegative cost function $J:H\to[0,\infty)$. For the time being, we only assume that J is a smooth function. In particular, J is allowed to have several local minima.

Assumption 3.1: J is continuously differentiable and its derivative satisfies the Lipschitz condition

$$||\nabla J(x)-\nabla J(x')|| \leq K||x-x'||, \qquad \forall x,x' \in H , \tag{3.1}$$

where K is some nonnegative constant.

Assumption 3.2: The updates $s_\ell^i(n)$ of each processor satisfy

$$\frac{\partial J}{\partial x_\ell}(x^i(n))E[s_\ell^i(n)|F_n] \leq 0, \qquad \text{a.s.}, \quad \forall i,\ell,n. \tag{3.2}$$ [5]

This assumption states that each component of each processor's updates is in a descent direction, when conditioned on the past history of the algorithm and it is satisfied by Examples I-IV. A slightly weaker version, under which our results remain true would be

$$\nabla J(x^i(n))\Phi^i(n)E[s^i(n)|F_n] \leq 0, \qquad \text{a.s.}, \quad \forall i, \ell,n.$$

On the other hand, it can be shown that the condition

$$\nabla J(x^i(n))E[s^i(n)|F_n] \leq 0, \qquad \text{a.s.}$$

is not sufficient for proving convergence.

The next assumption is easily seen to hold for Examples I and II. For stochastic algorithms, it requires that the variance of the updates (and hence of any noise contained in them) goes to zero, as the gradient of the cost function goes to zero.

[5] In (3.2), if H_ℓ has dimension larger than 1, $\frac{\partial J}{\partial x_\ell}$ should be interpreted as a row vector. In general, the appropriate interpretation should be clear from the context.

Assumption 3.3: For some $K_o \geq 0$ and for all i,ℓ,n,

$$E\left[||s_\ell^i(n)||\right]^2 \leq - K_o E\left[\frac{\partial J}{\partial x_\ell} (x^i(n))s_\ell^i(n)\right] .$$

As a matter of verifying Assumption 3.3, one would typically check the validity of the slightly stronger condition

$$E\left[||s_\ell^i(n)||^2 \; |F_n\right] \leq - K_o \frac{\partial J}{\partial x_\ell} (x^i(n))E\left[s_\ell^i(n)|F_n\right] .$$

Also, using Lemma 2.1 (ii) and Assumption 3.3 we obtain

$$E\left[||s^i(n)||^2\right] \leq \sum_\ell^L E\left[||s_\ell^i(n)||^2\right] \leq - \frac{K_o}{\eta} \sum_{\ell=1}^L E\left[\frac{\partial J}{\partial x_\ell} (x^i(n))\Phi_\ell^i(n)s_\ell^i(n)\right]$$

$$= -\overline{K}_o E[\nabla J(x^i(n))\Phi^i(n)s^i(n)], \qquad (3.3)$$

where $\overline{K}_o = K_o/\eta \geq 0$. It turns out that (3.3) is all we need for our results to hold.

Our first convergence result states that the algorithm converges in a suitable sense, provided that the step-size employed by each processor is small enough and that the time between consecutive communications is bounded, and applies to Examples I and II. It should be noted, however, that Theorem 3.1 (as well as Theorem 3.2 later) does not prove yet convergence to a minimum or a stationary point of J. In particular, there is nothing in our assumptions that prohibits having $s^i(n)=0$, ∀i,n. Optimality is obtained later, using a few auxiliary and fairly natural assumptions (see Corollary 3.1).

Theorem 3.1: Let Assumptions 2.1, 2.2, 2.4, 3.1, 3.2, 3.3 hold. Suppose also that $\gamma^i(n) \geq 0$ and that $\sup_{i,n} \gamma^i(n) = \gamma_0 < \infty$. There exists a constant $\gamma^* > 0$ (depending on the constants introduced in the Assumptions) such that the inequality $0<\gamma_0\leq\gamma^*$ implies:

a) $J(x^i(n))$, $i=1,2,\dots,M$, as well as $J(y(n))$, converge almost surely, and to the same limit.

b) $\lim_{n\to\infty}(x^i(n)-x^j(n)) = \lim_{n\to\infty}(x^i(n)-y(n))=0, \quad \forall i,j,$

almost surely and in the mean square.

c) The expression

$$\sum_{n=1}^{\infty}\sum_{i=1}^{M}\gamma^i(n)\nabla J(x^i(n))E[s^i(n)|F_n] \tag{3.4}$$

is finite, almost surely. Its expectation is also finite.

Proof: See Appendix B.

Leaving technical issues aside, the idea behind the proof of the above (and the next) Theorem is rather simple: the difference between $y(n)$ and $x^i(n)$, for any i, is of the order of $B\gamma_0$, where B is proportional to a bound on communication delays plus the time between consecutive communications between processors. Therefore, as long as γ_0 remains small, $\nabla J(x^i(n))$ is approximately equal to $\nabla J(y(n))$; hence $s_\ell^i(n)$ (and consequently $\Phi^i(n)s^i(n)$) is approximately in a descent direction, starting from point $y(n)$. Therefore, iteration (2.14) is approximately the same as a centralized descent (pseudo-gradient) algorithm which is, in general convergent [16].

Decreasing Step-Size Algorithms

We now introduce an alternative set of assumptions. We allow the magnitude of the updates $s^i(n)$ to remain nonzero, even if $\nabla J(x^i(n))$ is zero (Examples III and IV). Such situations are common in stochastic approximation algorithms or in system identification applications. Since the noise is persistent, the algorithm can be made convergent only by letting the step-size $\gamma^i(n)$ decrease to zero. The choice $\gamma^i(n)=1/n$ is most commonly used in centralized algorithms and in the sequel we will assume that $\gamma^i(n)$ behaves like 1/n.

Since the step-size is decreasing, the algorithm becomes progressively slower as $n\to\infty$. This allows us to let the communications process become progressively slower as well, provided that it remains fast enough, when compared with the natural time scale of the algorithm, the latter being determined by the rate of decrease of the step-size. Such a possibility is captured by Assumption 2.3.

The next assumption, intended to replace Assumption 3.3, allows the noise to be persistent. It holds for Examples I-IV. As in Assumption 3.3, inequality (3.5) could be more naturally stated in terms of conditional expectations, but such a stronger version turns out to be unnecessary.

<u>Assumption 3.4</u>: For some K_1, $K_2 \geq 0$, and for all i,ℓ,n,

$$E\left[||s^i_\ell(n)||^2\right] \leq -K_1 E\left[\frac{\partial J}{\partial x_\ell}(x^i(n))s^i_\ell(n)\right] + K_2 \qquad (3.5)$$

<u>Theorem 3.2</u>: Let Assumptions 2.1, 2.3, 2.4, 3.1, 3.2, 3.4, hold and assume that for some $K_3 \geq 0$, $\gamma^i(n) \leq K_3/n$, $\forall n,i$. Then, conclusions (a),(b),(c), of Theorem 3.1 remain valid.

<u>Proof</u>: See Appendix B.

Theorem 3.2 remain valid if (3.5) is replaced by the much weaker assumption

$$E\left[||s^i(n)||^2\right] \leq K_0 E\left[J(x^i(n))\right] - K_1 E\left[\nabla J(x^i(n))\Phi^i(n)s^i(n)\right] + K_2 . \qquad (3.6)$$

The proof may be found in [19] and is significantly more complicated. Theorems 3.1, 3.2 also remain valid even if Assumptions 3.2, 3.3 or 3.4 hold after some finite time n_o, but are violated earlier. We only need to assume that $s^i(k)$ has finite mean and variance, for $k \leq n_o$.

We continue with a corollary which shows that, under reasonable conditions, convergence to a stationary point or a global optimum may be guaranteed. We only need to assume that away from stationary points some processor will make a positive improvement in the cost function. Naturally, we only require the processors to make positive improvements at times that they are not idle.

Corollary 3.1: Suppose that for some $K_4>0$, $\gamma^i(n) \geq K_4/n$, $\forall n,i$. Assume that J has compact level sets and that there exist continuous functions g^i_ℓ: $H \to [0,\infty)$ such that

$$\frac{\partial J}{\partial x_\ell}(x^i(n))E[s^i_\ell(n)|F_n] \leq -g^i_\ell(x^i(n)), \quad \forall n \in T^i_\ell . \tag{3.7}$$

We define g: $H \to [0,\infty)$ by $g(x) = \sum_{i=1}^{M} \sum_{\ell=1}^{L} g^i_\ell(x)$ and we assume that any point $x \in H$ satisfying $g(x)=0$ is a stationary point of J. Finally, suppose that the difference between consecutive elements of T^i_ℓ is bounded, for any i,ℓ such that $T^i_\ell \neq \emptyset$. Then,

a) Under the Assumptions of either Theorem 3.1 or 3.2,

$$\liminf_{n\to\infty} ||\nabla J(x^i(n))||=0, \quad \forall i, \quad \text{a.s.} \tag{3.8}$$

b) Under the Assumptions of Theorem 3.1 and if (for some $\varepsilon>0$) $\gamma^i(n) \geq \varepsilon$, $\forall i,n$, we have

$$\lim_{n\to\infty} ||\nabla J(x^i(n))||=0, \quad \forall i, \quad \text{a.s.} \tag{3.9}$$

and any limit point of $\{x^i(n)\}$ is a stationary point of J.

c) Under the Assumptions of either Theorem 3.1 or 3.2 and if every point satisfying $g(x)=0$ is a minimizing point of J (this is implicitly assuming that all stationary point of J are minima), then

$$\lim_{n\to\infty} J(x^i(n)) = \inf_{x\in H} J(x) \quad . \tag{3.10}$$

Proof: See Appendix B.

We now discuss the above corollary and apply it to our examples. Notice first that the assumption on T^i_ℓ states that, for each component ℓ, the time between successive computations of s^i_ℓ is bounded, for any computing processor i for that component. Such a condition will be always met in practice. The assumption $\gamma^i(n) \geq K_4/n$ may be enforced without the processor having access to a global clock. For example, apart from the trivial case of constant step-size, we may let $\gamma^i(n) = 1/t^i_n$, where t^i_n is the number of times, before time n, that processor i has performed a computation.

For Examples I and III, (3.7) holds with $g^i_i(x)$ a constant multiple of $(\partial J/\partial x_i)^2$; for Examples II and IV, it holds with $g^i(x)$ a constant multiple of $||\nabla J(x)||^2$. We may conclude that Corollary 3.1 applies and proves convergence for Examples I-IV.

We close this section by pointing out that our results remain valid if we model the combining coefficients, the transmission and reception times as random variables defined on the same probability space (Ω, F, P), subject to some restrictions. Notice that such a generalization allows the processors to decide when and where to transmit based on information related to the progress of the algorithm. Hence, for stochastic algorithms, we have to avoid the possibility that a processor judiciously chooses to transmit at those times that it receives a "bad measurement" $s^i(n)$, or somehow ensure that the long-run outcome is independent of such decisions. It turns out that one only needs to assume that, for any $i\in\{1,\ldots,M\}$ and any $m\leq n$, the random variables $\Phi^i(m)$ and $s^i(n)$, are conditionally independent given the history of the algorithm up to time n [19]. This assumption holds if communications and the combining coefficients are independent of the noise in the computations (this is true, in particular, for deterministic algorithms), as well as for the specialization case because $\Phi^i(n)$ turns out to be deterministic and a priori known, even if the communication times are random.

4. EXTENSIONS, APPLICATIONS AND CONCLUSIONS

A main direction along which our results may be extended is in analyzing the convergence of distributed algorithms with decreasing step-size and with correlated noise, for which the pseudo-gradient assumption fails to hold. Such algorithms arise frequently, for example in system identification. Very few global convergence results are available, even for the centralized case [17]. However, as in the centralized case an ordinary differential equation (ODE) may be associated with such algorithms, which may be used to prove local convergence subject to an assumption that the algorithm returns infinitely often to a bounded region [11,12,19].

Another issue, arising in the case of constant step-size algorithms, concerns the choice of a step-size which will guarantee convergence. We may trace the steps in the proof of Theorem 3.1 and find some bounds on γ_0 so as to ensure convergence, but these boundswill not be particularly tight. For a version of a distributed deterministic gradient algorithm, tighter bounds have been obtained in [19] which quantify the notion that the frequency of communications between different processors should in some sense reflect the degree of coupling inherent in the optimization problem.

It should be clear that the bounds on the time between successive communications and delays are imposed so as to guarantee that processors are being informed, without excessive delays, about changes in the state of computation of other processors. The same effect, however, could be accomplished by having each processor monitor its state and inform the others only if a substantial change occurs. It seems that such an approach could lead to some savings in the number of messages being exchanged. However, more research is needed on this issue and, in particular, the notion of a "substantial change" in the state of computation of a processor needs to be made more precise.

A final issue of interest is the rate of convergence of distributed algorithms. With perfect synchronization, the rate of convergence is the same as for their centralized counterparts. Asynchronism should be expected to bring about some deterioration. We may then ask how much asynchronism may be tolerated before such deterioration becomes significant. For decreasing step-size stochastic algorithms, as long as communications do not become too infrequent, it is safe to conjecture that the convergence rate will not deteriorate at all; still, there would be some deterioration in the transient performance of such algorithms which is also worth investigating.

Concerning possible applications, there are three broad areas that come to mind. There is first the area of parallel computation, where an asynchronous algorithm could avoid several types of bottlenecks [10]. Then, there is the area data communication networks in which there has been much interest for distributed algorithms for routing and flow control [6,9]. Such algorithms resemble the ones that we have analyzed and our methods of analysis, with a few modifications, are applicable [18]. Finally, certain common algorithms for system identification or adaptive filtering fall into the framework of decreasing step-size stochastic algorithms and our approach may be used for analyzing the convergence of their distributed versions [19]. Our results may not be applicable without any modifications or refinements to such diverse application areas. Nevertheless, our analysis indicates what kind of results should be expected to hold and provides tools for proving them.

As a conclusion, the natural distributed asynchronous versions of a large class of deterministic and stochastic optimization algorithms retain the desirable convergence properties of their centralized (or synchronous) counterparts, under mild assumptions on the timing of communications and computations and even in the presence of communication delays. It appears that there are many and promising applications of such algorithms in fields as diverse as parallel computation, data communication networks and distributed signal processing.

REFERENCES

1. 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, N.C., 1960, pp. 34-104.

2. Avriel, M., Nonlinear Programming, Prentice-Hall, Englewood Cliffs, N.J., 1976.

3. Baudet, G.M., "Asynchronous Iterative Methods for Multiprocessors," Journal of the ACM, Vol. 25, No. 2, 1978, pp. 226-244.

4. Bertsekas, D.P., "Distributed Dynamic Programming," IEEE Transactions on Automatic Control, Vol. AC-27, No. 3, 1982, pp. 610-616.

5. Bertsekas, D.P., "Distributed Asynchronous Computation of Fixed Points," Mathematical Programming, Vol. 27, 1983, pp. 107-120.

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, 1982, pp. 615-643.

7. Bertsekas, D.P., J.N. Tsitsiklis and M. Athans, "Convergence Theories of Distributed Iterative Processes: A Survey," submitted to SIAM Review, 1984.

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

9. Gallager, R.G., "A Minimum Delay Routing Algorithm Using Distributed Computation," IEEE Transactions on Communications, Vol. COM-25, No. 1, 1977, pp. 73-85.

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

11. Kushner, H.J. and D.S. Clark, Stochastic Approximation Methods for Constrained and Unconstrained Systems, Applied Math. Series, No. 26, Springer Verlag, Berlin, 1978.

12. Ljung, L., "Analysis of Recursive Stochastic Algorithms," IEEE Transactions on Automatic Control, Vol. AC-22, No. 4, 1977, pp. 551-575.

13. Ljung, L., and T. Soderstrom, Theory and Practice of Recursive Identification, MIT Press, Cambridge, MA, 1983.

14. Meyer, P.A., Probability and Potentials, Blaisdell, Waltham, MA, 1966.

15. Papadimitriou, C.H. and J.N. Tsitsiklis, "On the Complexity of Designing Distributed Protocols," Information and Control, Vol. 53, No. 3, June 1982, pp. 211-218.

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

17. Solo, V., "The Convergence of AML," IEEE Transactions on Automatic Control, Vol. AC-24, 1979, pp. 958-962.

18. Tsitsiklis, J.N., and D.P. Bertsekas, "Distributed Asynchronous Optimal Routing for Data Networks," Proceedings of the 23d Conference on Decision and Control, Las Vegas, Nevada, December 1984.

19. Tsitsiklis, J.N., "Problems in Decentralized Decision Making and Computation," Ph.D. Thesis, Dept. of Electrical Engineering and Computer Science, MIT, Cambridge, MA, 1984.

20. Yao, A.C., "Some Complexity Questions Related to Distributed Computing," Proceedings of the 14th STOC, 1979, pp. 209-213.

APPENDIX A

Proof of Lemma 2.1: We only need to prove the Lemma for each component separately, since (2.1) corresponds to a decoupled set of linear systems. We may therefore assume that there is only one component; so, the subscript ℓ will be omitted and $\Phi^{ij}(n|k)$ becomes a scalar.

The coefficient $\Phi^{ij}(n|k)$ being the impulse response of the linear system (2.1) is determined by the following "experiment": let us fix a processor j and some time k; let $x^h(1)=0$, $\forall h$, $s^h(m)=0$, $\forall h,m$, unless if h=j and m=k in which case we let $\gamma^j(k)s^j(k)=v$, some nonzero element of H. For all times n and for all processors i, $x^i(n)$ will be a scalar multiple of v. This proportionality factor is precisely equal to $\Phi^{ij}(n|k)$. Since this experiment takes place in a one-dimensional subspace of H, we may assume -without loss of generality- that H is one-dimensional and that v=1. We then have, for the above "experiment", $\Phi^{ij}(n|k) = x^i(n)$, $\forall i,n$. A trivial induction based on (2.1) and using (2.2) shows that $x^i(n)\geq 0$, $\forall i,n$, which proves (2.7).

For inequality (2.8) we consider a different "experiment": let us fix some time k and let $x^h(1)=0$, $\forall h$, $s^h(m)=0$, $\forall m,h$, unless if m=k in which case we let $\gamma^h(k)s^h(k)=1$, $\forall h$. (H is still assumed one-dimensional). We then use (2.1) and (2.3) to show by a simple inductive argument that $x^i(n)\leq 1$, $\forall i,n$ which concludes the proof of part (i).

For the proof of the remaining parts of the Lemma we first perform a reduction to a simpler case. It is relatively easy to see that we may assume, without loss of generality, that communication delays are nonzero. For example, we could redefine the time variable so that one time unit for the old time variable corresponds to two time units for the new one and so that any message that had zero delay for the original description has unit delay for the new description. If any of the Assumptions 2.1, 2.2, 2.3, 2.4 holds in terms of the old time variable, it also remains true with the new one.

Next, we perform a reduction to the case where all messages have zero delay, as follows: for any $(i,j)\in E$, we introduce a finite set of dummy processors, between i and j (see Fig. A.1) which act as buffers. Any message from i to j is first transmitted to a buffer processor (with zero delay) which holds it for time equal to the desired delay and then transmits it to j, again with zero delay. (So, whenever a buffer processor h receives a message, it lets $a^{hi}(n)=1$). The buffer processor which is to be employed for any particular message is determined by a round-robin rule. Note that for each pair $(i,j)\in E$ the number of buffer processors that needs to be introduced is equal to the maximum communication delay for messages from i to j, which has been assumed finite.[6] Note also that the buffer processors are non-computing ones and have in-degree equal to 1.

It is easy to see that if Assumptions 2.2, 2.3 are valid for the original description of the algorithm, they are also valid for the above introduced augmented description, possibly with a different choice of constants. Assumption 2.1 also remains valid except for part c(iii) which may be violated. (If in Figure A.1 processor j had originally in-degree equal to 1, it now has in-degree equal to 4, but there is nothing in our assumptions that guarantees that $a^{jj}(n)\geq\alpha$, $\forall n$). We have, nevertheless, the following condition:

[6] This procedure is equivalent to a state augmentation for the linear system (2.2).

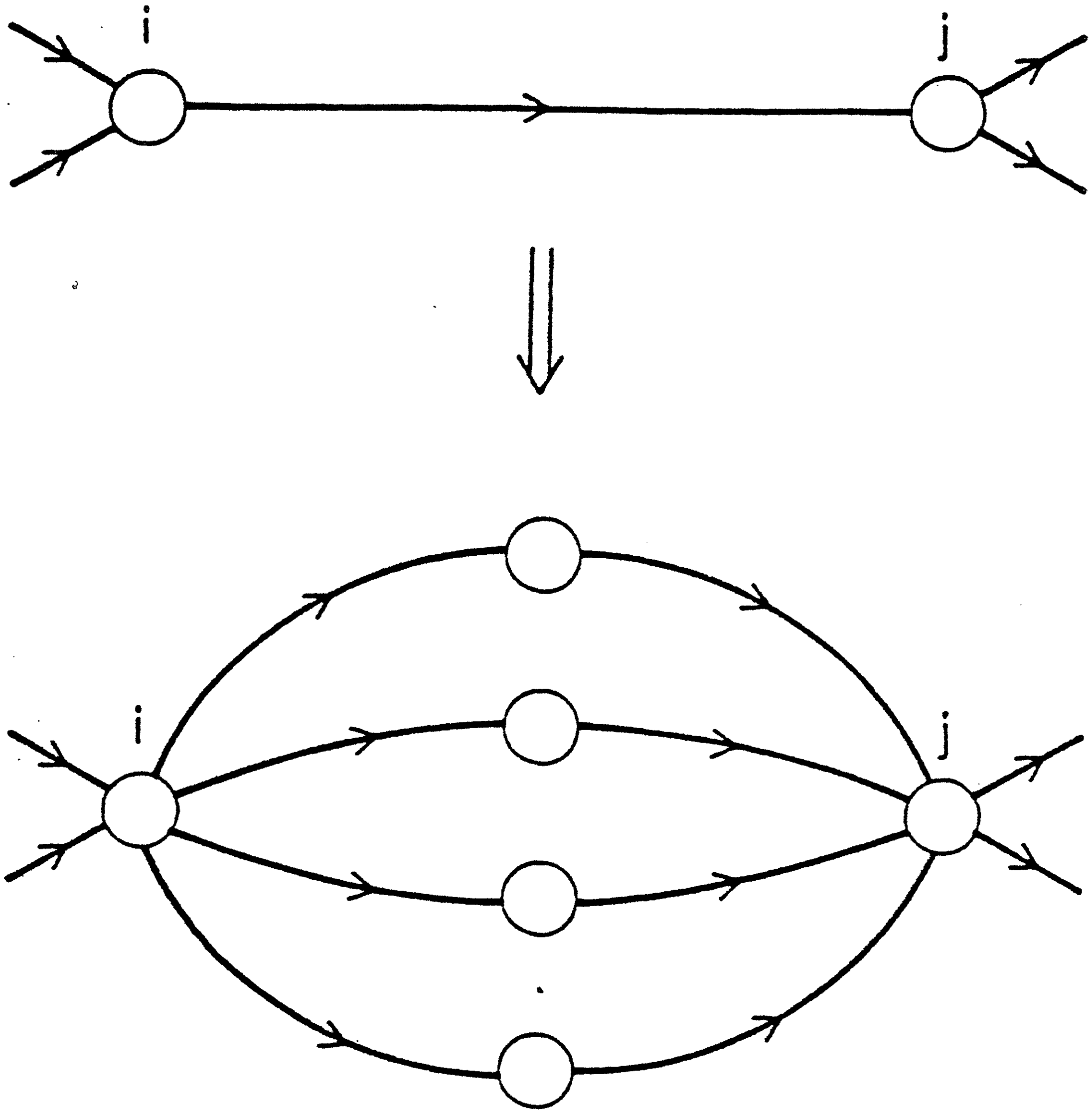

Figure 1: Reduction to the zero delay case.

Assumption A.1: For any processor i with in-degree larger than 1, either

1. Assumption 2.1c(iii) holds, or

2. All predecessors of i are non-computing, have in-degree 1 and a common predecessor.

From now on, we assume, without loss of generality, that communication delays are zero, provided that we replace Assumption 2.1c(iii) by Assumption A.1.

Let $A(n)$, $\Phi(n|k)$ be the matrices with coefficients $a^{ij}(n)$, $\Phi^{ij}(n|k)$, respectively. Because of the zero delay assumption it is easy to see that $\Phi(n|k) = \prod_{m=k+1}^{n-1} A(m)$, $n>k+1$.[7]

We first prove the desired results under Assumptions 2.1 and 2.2. By combining Assumptions 2.1c(i) and 2.2, note that there exists a constant B such that, for any $(i,j)\in E$ and any time interval I of length B, there exists some $t\in I$ such that $a^{ji}(t)\geq\alpha>0$. (These are times that i communicates to j.)

Let us fix a computing processor j and some time k. By relabelling, let us assume that j=1. We will show by induction (with respect to a particular numbering of the processors) that for any processor i, there exists some $\alpha_i>0$, independent of k, such that $\Phi^{i1}(n|k)\geq\alpha_i$, for all $n\in[k+(i-1)B+1, k+MB]\triangleq I_i$.

To start the induction, consider first the case i=1 and notice that

$$\Phi^{11}(n|k)\geq \prod_{t=k+1}^{n-1} a^{11}(t)\geq \alpha^{n-k-1}\geq \alpha^{MB} \triangleq \alpha_1>0, \quad \forall n\in I_i .$$

[7] Since each A(n) is a "stochastic matrix (nonnegative entries, each row sums to 1), questions of convergence of $\Phi(n|k)$ are equivalent to questions about the long-run behavior of a finite (time-varying) Markov chain.

Suppose now that a subset S of the processors has been numbered $\{1,\ldots,i-1\}$, for some $i\geq 2$, and that the induction hypothesis has been proved for all processors in S. We show that it is always possible to find a new processor in V/S, rename it to i and prove the induction hypothesis for i as well.

Consider the set Q of processors $q\notin S$ such that $(p,q)\in E$, for some $p\in S$. If $Q=\phi$, then S is the set of all processors (because of Assumption 2.1b) and we are done. If not, we choose one processor from Q, and rename it to i, subject to the following restriction: we choose a processor with in-degree more than one only if no processor with in-degree equal to one belongs to Q.

We now prove the induction hypothesis for processor i. Let $h\in S$ be some predecessor of i, belonging to S. Then, $h<i$ and, by the induction hypothesis, $\Phi^{h1}(n|k)\geq\alpha_h>0$, $\forall n\in I_h\subset I_{i-1}$. Moreover, for some $t\in[k+(i-2)B+1,\ldots,k+(i-1)B]$ we have $a^{ih}(t)\geq\alpha$ and consequently, $\Phi^{i1}(t+1|k)\geq\alpha\alpha_h$.

We first suppose that i has in-degree 1. We prove by induction on n, for $n\in[t+1,\ldots,k+MB]\supset I_i$ that $\Phi^{i1}(n|k)\geq\alpha\alpha_h\overset{\Delta}{=}\alpha_i>0$. Indeed,

$$\Phi^{i1}(n+1|k) = a^{ih}(n)\Phi^{h1}(n|k) + a^{ii}(n)\Phi^{i1}(n|k)\geq$$

$$\geq\min\{\Phi^{h1}(n|k),\Phi^{i1}(n|k)\}\geq\min\{\alpha_h,\alpha\alpha_h\} = \alpha\alpha_h\ .$$

Suppose now that i has in-degree more than 1 and that Assumption 2.1c(iii) holds. Then,

$$\Phi^{i1}(n|k)\geq\left[\prod_{m=t+1}^{n-1} a^{ii}(m)\right]\Phi^{i1}(t+1|k)\geq\alpha^{n-t}\alpha_h\geq\alpha^{MB}\alpha_h\overset{\Delta}{=}\alpha_i>0,\quad\forall n\in I_i\ .$$

The last possibility is that i has in-degree more than 1 (hence all processors in Q have in-degree more than 1) and Assumption 2.1c(iii) fails. Then the set of predecessors of i (denoted by U) has a single common predecessor, denoted by j. Since $i\in Q$, some $h\in U$ must belong to S. Since any $h\in U$ is non-computing, we have $h\neq 1$ and its predecessor j must belong to S. Now, for any $p\in U$, p does not belong to Q (since it has in-degree 1) and therefore, $p\in S$. We conclude that all predecessors of i belong to S $(U\subset S)$. We now perform an easy induction on n, for $n\in[t+1,\dots,k+MB]$ to show that $\Phi^{i1}(n|k)\geq\alpha \min_{h\in U} \{\alpha_h\} \triangleq \alpha_i>0$. Indeed,

$$\Phi^{i1}(n+1|k) = \sum_{h\in U\cup\{i\}} a^{ih}(n)\Phi^{h1}(n|k) \geq \min_{h\in U\cup\{i\}} \Phi^{h1}(n|k) \geq$$

$$\geq \min\{\alpha_i, \min_{h\in U} \{\alpha_h\}\} = \alpha_i \quad .$$

This completes our inductive argument.

We may now conclude that $\Phi(k+MB|k)$ is a stochastic matrix with the property that all entries in some column (corresponding to any computing processor) are positive and bounded away from zero by a constant $\bar{\alpha}>0$ which does not depend on k. We combine this fact with Lemma A.1 below to conclude that the assertions of Lemma 2.1 are true.

Lemma A.1: Consider a sequence $\{D_n\}$ of nonnegative matrices with the properties that:

(i) Each row sums to 1.

(ii) For some $\bar{\alpha}>0$ and for some column (say the first one), all entries of D_n, for any n, in that column are larger or equal than $\bar{\alpha}$.

Then,

a) $\overline{D} = \lim_{n\to\infty} \prod_{k=1}^{n} D_k$ exists.

b) All rows of $\overline{D}$ are identical.

c) The entry in the first column of $\overline{D}$ is bounded below by $\overline{\alpha}$.

d) Convergence to $\overline{D}$ takes place at the rate of a geometric progression.

Proof of Lemma A.1: Given any vector $x=(x_1,\ldots,x_M)$ we decompose it as $x=y+ce$, where c is a scalar, e is the vector with all entries equal to 1 and y has one zero entry and all other entries are nonnegative. (So c equals the minimum of the components of x.

Let $x(n) = \prod_{k=1}^{n} D_k x(0)$, $\forall n$ and $x(n)=y(n)+c(n)e$. It is easy to see that $||y(n+1)||_\infty \leq (1-\overline{\alpha})||y(n)||_\infty$, where $||\cdot||_\infty$ denotes the max-norm on R^M, which shows that y(n) converges geometrically to zero. Moreover, $c(n)\leq c(n+1)\leq c(n) + ||y(n)||_\infty$, which shows that c(n) also converges geometrically to some $\overline{c}$. Hence, x(n) converges geometrically to $\overline{c}e$. Since this is true for any $x(0)\in R^n$, parts (a),(b),(d) ,of the Lemma follow. Part (c) is proved by an easy induction, for the finite products of the D_k's and, therefore, it holds for $\overline{D}$ as well. □

We now consider the case where Assumption 2.2 is replaced by 2.3. The key observation here is that during an interval of the form $[B_1 n^\beta, B_1(n+1)^\beta]$ a bounded number of messages is transmitted; hence, A(k)=I, except for a bounded number of times in that interval. If we redefine the time variable so that time is incremented only at communication times, we have reduced the problem to the case of Assumption 2.2. The only difference, due to the change of the time variable, is in the rate of convergence. Under Assumption 2.2, $||\Phi(n|k)-\Phi(k)||$ decreases by a constant factor during intervals of constant length. This implies that, under Assumption 2.3, $||\Phi(n|k)-\Phi(k)||$ decreases by a constant factor during intervals of the form $[B_1 t^{\beta}, B_1(t+c)^{\beta}]$, for some appropriate constant c. Therefore, if $B_1 t^{\beta}=k$ and $B_1(t+m)^{\beta}=n$, we have $||\Phi(n|k)-\Phi(k)|| \leq Bd_o^{m/c}$, for some $B\geq 0$, $d_o \in [0,1)$. Eliminating t and solving for m, we obtain

$$\left(\frac{k}{B_1}\right)^{1/\beta} + m = \left(\frac{n}{B_1}\right)^{1/\beta} ,$$

which finally yields

$$||\Phi(n|k)-\Phi(k)|| \leq Bd_o^{\left[\left(\frac{n}{B_1}\right)^{1/\beta} - \left(\frac{k}{B_1}\right)^{1/\beta}\right]/c} .$$

Let $\delta=1/\beta$ and $d=d_o^{\frac{1}{cB_1^{1/\beta}}}$, to recover the desired result. ■

APPENDIX B

This Appendix contains the proofs of the results of Section 3.

Remark on Notation: In the course of the proofs in this section, we will use the symbol A to denote non-negative constants which are independent of n, γ_0, $\gamma^i(n)$, $x^i(n)$, $s^i(n)$ etc., but which may depend on the constants introduced in the various assumptions (that is, M, L, K, K_0, B_0, B_1, α, etc.). When A appears in different expressions, or even in different sides of the same equality (or inequality), it will not necessarily represent the same constant. (With this convention, an inequality of the form $A + 1 \leq A$ is meaningful and has to be interpreted as saying that A+1, where A is some constant, is smaller than some other constant, denoted again by A.) This convention is followed so as to avoid the introduction of unnecessarily many symbols.

Without loss of generality, we will assume that the algorithm is initialized so that $x^i(1)=0$, $\forall i$. In the general case where $x^i(1)\neq 0$, we may think of the algorithm as having started at time 0, with $x^i(0)=0$; then, a random update $s^i(0)$ sets $x^i(1)$ to a nonzero value. So, the case in which the processors initially disagree may be easily reduced to the case where they initially agree.

Note that we may define $\bar{s}^i(n) = \frac{\gamma^i(n)}{\gamma_0} s^i(n)$ and view $\bar{s}^i(n)$ as the new step with step-size γ_o. It is easy to see that Assumptions (3.2) and (3.3) also hold for $\bar{s}^i(n)$. For these reasons, no generality is lost if we assume that $\gamma^i(n) = \gamma_0$, $\forall n$ and this is what we will do.

Let us define

$$b(n) = \sum_{i=1}^{M} ||s^i(n)|| \tag{B.1}$$

and note that

$$b^2(n) \leq M \sum_{i=1}^{M} ||s^i(n)||^2 \leq Mb^2(n) .$$

Using (2.6), (2.13) and Lemma 2.1 (iii), we obtain

$$||y(n)-x^i(n)|| \leq \sum_{k=1}^{n-1} \sum_{j=1}^{M} \gamma_0 ||\Phi^j(k)-\Phi^{ij}(n|k)|| \; ||s^j(k)|| \leq$$

$$\leq A\gamma_0 \sum_{k=1}^{n-1} d^{n-k} b(k) \; . \tag{B.2}$$

From a Taylor series expansion for J we obtain

$$J(y(n+1)) = J\left(y(n)+\gamma_0 \sum_{i=1}^{M} \Phi^i(n)s^i(n)\right) \leq$$

$$\leq J(y(n)) + \gamma_0 \nabla J(y(n)) \sum_{i=1}^{M} \Phi^i(n)s^i(n) + A||\gamma_0 \sum_{i=1}^{M} \Phi^i(n)s^i(n)||^2$$

$$\leq J(y(n)) + \gamma_0 \nabla J(y(n)) \sum_{i=1}^{M} \Phi^i(n)s^i(n) + A\gamma_0^2 b^2(n) \; . \tag{B.3}$$

Assumption 3.2 is in terms of $\nabla J(x^i(n))$, whereas above we have $\nabla J(y(n))$. To overcome this difficulty, we use the Lipschitz continuity of the derivative of J and invoke (B.2) to obtain

$$||\nabla J(y(n)) \sum_{i=1}^{M} \Phi^i(n)s^i(n) - \sum_{i=1}^{M} \nabla J(x^i(n))\Phi^i(n)s^i(n)|| \leq$$

$$\leq A \sum_{i=1}^{M} ||y(n)-x^i(n)|| \; ||s^i(n)|| \leq$$

$$\leq \gamma_0 A \sum_{k=1}^{n-1} d^{n-k} b(k) \sum_{i=1}^{M} ||s^i(n)|| = \gamma_0 A \sum_{k=1}^{n-1} d^{n-k} b(k)b(n) \leq$$

$$\leq \gamma_0 A \sum_{k=1}^{n-1} d^{n-k} [b^2(k)+b^2(n)] \; . \tag{B.4}$$

Let us define

$$G^i(n) = -\nabla J(x^i(n))\Phi^i(n)s^i(n) \; , \tag{B.5}$$

$$G(n) = \sum_{i=1}^{M} G^i(n), \tag{B.6}$$

and note that Assumption 3.2 implies that $E[G(n)] \geq 0$. We now rewrite inequality

(B.3) using (B.4) to replace the derivative term, to obtain:

$$J(y(n+1)) \le J(y(n)) - \gamma_0 G(n) + A\gamma_0^2 b^2(n) + A\gamma_0^2 \sum_{k=1}^{n-1} d^{n-k}[b^2(k)+b^2(n)] \le$$

$$\le J(y(n)) - \gamma_0 G(n) + A\gamma_0^2 \sum_{k=1}^{n} d^{n-k} b^2(k) \ . \tag{B.7}$$

Assumption 3.3 implies that [cf. inequality (3.3)]

$$E[b^2(k)] \le AE[G(k)] \ . \tag{B.8}$$

Taking expectations in (B.7) and using (B.8) we obtain

$$E[J(y(n+1))] \le E[J(y(n))] - \gamma_0 E[G(n)] + A\gamma_0^2 \sum_{k=1}^{n} d^{n-k} E[G(k)] \ . \tag{B.9}$$

We then sum (B.9) for different values of n, to obtain

$$0 \le E[J(y(n+1))] \le E[J(y(1))] + \sum_{k=1}^{n} \left[A \frac{\gamma_0^2}{1-d} - \gamma_0 \right] E[G(k)] \ . \tag{B.10}$$

We now let $\gamma^* = \frac{1-d}{2A}$, where A is the constant of inequality (B.10), and assume that $0<\gamma_0 \le \gamma^*$. Then, inequality (B.10) implies

$$\gamma_0 \sum_{k=1}^{n} E[G(k)] \le 2E[J(y(1))], \ \forall n, \tag{B.11}$$

and letting n tend to infinity,

$$\gamma_0 \sum_{k=1}^{\infty} E[G(k)] < \infty \ . \tag{B.12}$$

By Assumption 3.2, $E[G(k)|F_k]\geq 0$, $\forall k$; we may apply the monotone convergence theorem to (B.12) and obtain

$$E\left[\sum_{k=1}^{\infty} E[G(k)|F_k]\right] = \sum_{k=1}^{\infty} E[G(k)] <\infty \tag{B.13}$$

which implies

$$\sum_{k=1}^{\infty} E[G(k)|F_k]<\infty, \quad \text{a.s.} \tag{B.14}$$

From (B.14) we obtain

$$\sum_{n=1}^{\infty} \nabla_{\ell} J(x^i(n))E[\Phi_{\ell}^i(n)s_{\ell}^i(n)|F_n] >-\infty, \qquad \text{a.s.}$$

Now use the fact (Lemma 5.2.1 (ii), inequality (2.9)) that $\Phi_{\ell}^i(n)\geq\eta>0$, for any computing processor i for component ℓ. This implies that

$$\sum_{k=1}^{\infty} \nabla_{\ell} J(x^i(n))E[s_{\ell}^i(n)|F_n]>-\infty, \quad \text{a.s.}$$

and establishes part (c) of the theorem.

Lemma B.1: Let X(n), Z(n) be non-negative stochastic processes (with finite expectation) adapted to $\{F_n\}$ and such that

$$E[X(n+1)|F_n]\leq X(n) + Z(n), \tag{B.15}$$

$$\sum_{n=1}^{\infty} E[Z(n)]<\infty \quad . \tag{B.16}$$

Then X(n) converges almost surely, as $n\to\infty$.

Proof of Lemma 5.3.1: By the monotone convergence theorem and (5.3.23) it follows that $\sum_{n=1}^{\infty} Z_n < \infty$, almost surely. Then, Lemma 5.3.1 becomes the same as Lemma 4.C.1 in [13, p.453], which in turn is a consequence of the supermartingale convergence theorem [14]. □

Now let A be the constant in the right hand side of (B.7) and let

$$Z(n) = A\gamma_0^2 E\left[\sum_{k=1}^{n} d^{n-k} b^2(k) \,|\, F_n\right]. \tag{B.17}$$

Then, $Z(n) \geq 0$ and by (B.8)

$$E[Z(n)] \leq A \sum_{k=1}^{n} d^{n-k} E[G(k)] \ . \tag{B.18}$$

Therefore,

$$\sum_{n=1}^{\infty} E[Z(n)] \leq A \sum_{n=1}^{\infty} \sum_{k=1}^{n} d^{n-k} E[G(k)] =$$

$$= A \frac{1}{1-d} \sum_{k=1}^{\infty} E[G(k)] < \infty \ , \tag{B.19}$$

where the last inequality follows from (B.12) Therefore, Z(n) satisfies (B.16). We take the conditional expectation of (B.7), given F_n. Note that J(y(n)) is F_n-measurable and that $E[G(n)|F_n] \geq 0$. Therefore Lemma B.1 applies and J(y(n)) converges almost surely.

Using Assumption 3.3 once more, together with (B.12),

$$E\left[\sum_{k=1}^{\infty} b^2(k)\right] \leq A \sum_{k=1}^{\infty} E[G(k)] < \infty \tag{B.20}$$

which implies that $b(k)$ converges to zero, almost surely. Recall (B.2) to conclude that $y(n)-x^i(n)$ converges to zero, almost surely. Also, by squaring (B.2), taking expectations and using the fact that $E[b^2(k)]$ converges to zero we conclude that $E[||y(n)-x^i(n)||^2]$ also converges to zero, and this proves part (b) of the Theorem.

Now, let us use Assumption 3.1 and a second order expansion of J to obtain, for any $a \in R$

$$0 \leq J(x-a\nabla J(x)) \leq J(x) - aA_1||\nabla J(x)||^2 + a^2A_2||\nabla J(x)||^2 , \tag{B.21}$$

where A_1, A_2 are positive constants not depending on a. Assuming that a was chosen small enough, we may use (B.21) and the nonnegativity of J to conclude

$$||\nabla J(x)||^2 \leq AJ(x), \quad \forall x \in H. \tag{B.22}$$

Since $J(y(n))$ converges, it is bounded; hence $\nabla J(y(n))$ is also bounded, by (B.22). We then use the fact that $y(n)-x^i(n)$ converges to zero, to concludes that $J(x^i(n))-J(y(n))$ also converges to zero. This proves part (a) and concludes the proof of the Theorem. ■

In the following Lemma we bound certain infinite series by corresponding infinite integrals. This is justified as long as the integrand cannot change by more than a constant factor between any two consecutive integer points. For notational convenience, we use $c(n|k)$ to denote $d^{n^\delta - k^\delta}$, where d and δ are as in Lemma 2.1(iv).

<u>Lemma B.2</u>: The following hold:

$$\sum_{n=1}^{\infty} \sum_{m=n}^{\infty} \frac{1}{m} \frac{1}{n} c(m|n) < \infty, \tag{B.23}$$

$$\lim_{m\to\infty} \sum_{k=1}^{m} \frac{m}{k^2}\; c(m|k) = \lim_{m\to\infty} \sum_{k=1}^{m} \frac{1}{k}\, c(m|k)=0, \tag{B.24}$$

$$\lim_{m\to\infty} \sum_{k=m}^{\infty} \frac{1}{k}\, c(k|m)=0. \tag{B.25}$$

<u>Proof of Lemma B.2</u>: Let $t^{\delta}=y$; then, $t=y^{1/\delta}$ and $dt=(1/\delta)y^{\frac{1}{\delta}-1}dy$. Therefore,

$$\int_{t=s}^{\infty} \frac{1}{t}\, d^{t^{\delta}-s^{\delta}} dt = \frac{1}{\delta} \int_{y=s^{\delta}}^{\infty} d^{y-s^{\delta}} \frac{1}{y}\, dy \leq \frac{1}{\delta s^{\delta}} \int_{y=s^{\delta}}^{\infty} d^{y-s^{\delta}} dy \leq \frac{A}{s^{\delta}} \tag{B.26}$$

where A does not depend on s. Equation (B.25) follows. Since $\frac{1}{s}\;\frac{A}{s^{\delta}}$ is an integrable function of s, (B.23) follows as well.

The left hand side of (B.24) is bounded by

$$Am\int_{t=1}^{m} \frac{1}{t^{2}}\; d^{m^{\delta}-t^{\delta}} dt = Am\int_{y=1}^{m^{\delta}} \frac{1}{y^{2/\delta}}\, d^{m^{\delta}-y^{\delta}}\, \frac{1}{\delta}\; y^{\frac{1}{\delta}-1} dy =$$

$$= Am\int_{y=1}^{m^{\delta}} y^{-\frac{1}{\delta}-1}\; d^{m^{\delta}-y} dy \leq Am^{-\delta},$$

which converges to zero. The middle term in (B.24) is certainly smaller and converges to zero as well. □

<u>Proof of Theorem 3.2</u>: Using the same arguments as in the proof of Theorem 3.1, we may assume, without loss of generality, that $x^{i}(1)=0$ and that $\gamma^{i}(n)=1/n$, $\forall\, i,n$. (Otherwise we could define $\bar{s}^{i}(n) = n\gamma^{i}(n)s^{i}(n)$.)

We still use $c(n|k)$ to denote $d^{n^{\delta}-k^{\delta}}$. We define again $b(n)$, $G^{i}(n)$, $G(n)$ by (B.1), (B.5), (B.6), respectively, as in the proof of Theorem 5.3.1. Also, let

$$\phi(n|k) = \begin{cases} \frac{1}{n^2} + \sum_{m=1}^{n-1} \frac{1}{n}\frac{1}{m} \quad c(n|m), & n=k, \\ \frac{1}{n}\frac{1}{k} \; c(n|k), & n>k, \\ 0, \quad n<k. \end{cases} \tag{B.27}$$

By replicating the steps leading to inequality (B.7) in the proof of Theorem 3.1 and using Lemma 2.1(iv) and (B.27) we obtain, for some $A\geq 0$,

$$J(y(n+1)) \leq J(y(n)) - \frac{1}{n} G(n) + A \sum_{k=1}^{n} \phi(n|k) b^{2}(k), \;\; \forall n. \tag{B.28}$$

Taking expectations in (B.28), we have

$$E[J(y(n+1))] \leq E[J(y(n))] - \frac{1}{n} E[G(n)] + A \sum_{k=1}^{n} \phi(n|k) E[b^{2}(k)], \; \forall n. \tag{B.29}$$

and using Assumption 3.4,

$$E[J(y(n+1))] \leq E[J(y(n))] - \frac{1}{n} E[G(n)] + A \sum_{k=1}^{n} \phi(n|k)(E[G(k)]+1). \tag{B.30}$$

We then sum (B.30), for different values of n, to obtain

$$0 \leq E[J(y(n+1))] \leq E[J(y(1))] + A \sum_{m=1}^{n} \sum_{k=1}^{m} \phi(m|k) + \sum_{m=1}^{n} \left[\sum_{k=m}^{n} \phi(k|m) - \frac{1}{m} \right] E[G(m)]. \tag{B.31}$$

The definition (B.27) and (B.23) imply that the middle term in the right hand side of (B.31) is bounded. Moreover, using (B.27),

$$m \sum_{k=m}^{\infty} \phi(k|m) = \frac{1}{m} + \sum_{k=1}^{m-1} \frac{1}{k} c(m|k) + \sum_{k=m}^{\infty} \frac{1}{k} c(k|m),$$

which converges to zero, as $m \to \infty$, by (B.24) and (B.25). Therefore, for large enough m, $\sum_{k=m}^{n} \phi(k|m) - \frac{1}{m} \leq - \frac{1}{2m}$. It follows that $E[J(y(n))]$ is bounded. Inequality (B.31) and the above also imply that $\sum_{m=1}^{\infty} \frac{1}{m} E[G(m)] < \infty$ and part (c) of the Theorem follows, as in the proof of Theorem 3.1.

We now define

$$Z(n) = E\left[\sum_{k=1}^{n} \phi(n|k) b^2(k) | F_n \right]$$

and note that

$$\sum_{n=1}^{\infty} E[Z(n)] \leq A \sum_{k=1}^{\infty} \sum_{n=k}^{\infty} \phi(n|k) \left[E[G(k)] + 1 \right] \leq A + A \sum_{k=1}^{\infty} \frac{1}{k} E[G(k)] < \infty .$$

Taking conditional expectations in (B.28) (with respect to F_n) and using Lemma B.1, we conclude that $J(y(n))$ converges, almost surely.

We now turn to the proof of part (b). Using (3.5) and (B.22), we have

$$\begin{aligned} E\left[||s^i(n)||^2 \right] &\leq E\left[2A \nabla J(x^i(n)) \frac{1}{2} s^i(n)) \right] + A \leq \\ &\leq 4A^2 E\left[||\nabla J(x^i(n)||^2 \right] + \frac{1}{4} E\left[||s^i(n)||^2 \right] + A \leq \\ &\leq AE[J(x^i(n))] + \frac{1}{4} E\left[||s^i(n)||^2 \right] + A, \end{aligned} \tag{B.32}$$

which finally implies that

$$E[||s^i(n)||^2] \leq AE[J(x^i(n))] + A. \tag{B.33}$$

Now, using (B.22) once more,

$$J(x^i(n))-J(y(n))\leq ||\nabla J(y(n)||\cdot ||x^i(n)-y(n)|| + A||x^i(n)-y(n)||^2\leq$$

$$\leq \frac{1}{2}||\nabla J(y(n)||^2 + A||x^i(n)-y(n)||^2\leq AJ(y(n)) + A||x^i(n)-y(n)||^2. \tag{B.34}$$

Inequalities (B.33), (B.34) and the boundedness of $E[J(y(n))]$ (which is a consequence of (B.31)) yield

$$E[b^2(n)]\leq A+AE[||x^i(n)-y(n)||^2] . \tag{B.35}$$

Similarly with (B.2), we have

$$||y(n)-x^i(n)||\leq A\sum_{k=1}^{n-1}\frac{1}{k}c(n|k)b(k) \tag{B.36}$$

and

$$||y(n)-x^i(n)||^2\leq An\sum_{k=1}^{n-1}\frac{c^2(n|k)}{k^2}\ b^2(k)\leq An\sum_{k=1}^{n-1}\frac{c(n|k)}{k^2}b^2(k). \tag{B.37}$$

Therefore,

$$E[||y(n)-x^i(n)||^2]\leq \beta(n)\max_{1\leq k<n} E[b^2(k)], \tag{B.38}$$

where $\beta(n) = An\sum_{k=1}^{n-1}\frac{c(n|k)}{k^2}$ converges to zero, by (B.24). Using (B.35),

$$E[||y(n)-x^i(n)||^2]\leq A\beta(n)(1+\max_{k<n} E[||y(k)-x^i(k)||^2])$$

and since $\beta(n)$ converges to zero, it follows that $E[||y(n)-x^i(n)||^2]$ converges to zero as well. We also conclude from (B.35) that $\sup_n E[b^2(n)]<\infty$.

Let

$$D_k = \sum_{k^{1/\delta} \leq i < (k+1)^{1/\delta}} \frac{1}{i}\, b(1), \quad k \geq 1. \tag{B.39}$$

Using the fact that there exists an A such that $(k+1)^{1/\delta} - k^{1/\delta} \leq Ak^{(1/\delta)-1}$, $\forall k$, obtain from (B.39)

$$\sum_{k=1}^{\infty} E[D_k^2] \leq \sum_{k=1}^{\infty} \left[\sum_{k^{1/\delta} \leq i < (k+1)^{1/\delta}} \frac{1}{i} \right]^2 \sup_n E[b^2(n)] \leq$$

$$\leq A \sum_{k=1}^{\infty} \frac{1}{k^{2/\delta}} \left[(k+1)^{1/\delta} - k^{1/\delta} \right]^2 \leq A \sum_{k=1}^{\infty} \frac{1}{k^{2/\delta}}\, k^{(2/\delta)-2} < \infty. \tag{B.40}$$

If follows that D_k^2 converges to zero, almost surely. Consequently, so does D_k and $\sum_{k=1}^{n} d^{n-k} D_k$ as well. Let us fix some n, let N denote the largest integer such that $N \leq n^{\delta}$ and use (B.36) to obtain

$$||x^i(n) - y(n)|| \leq A \sum_{k=1}^{n-1} \frac{1}{k}\, c(n|k) b(k) \leq A \sum_{1 \leq k \leq n^{\delta}-1} \; \sum_{k^{1/\delta} \leq i < (k+1)^{1/\delta}} \left[\frac{1}{i}\, d^{n^{\delta} - i^{\delta}}\, b(i) \right] \leq$$

$$\leq A \sum_{1 \leq k \leq n^{\delta}-1} d^{N-(k+1)} \sum_{k^{1/\delta} \leq i < (k+1)^{1/\delta}} \frac{1}{i}\, b(i) \leq A \sum_{k=1}^{N} d^{N-k} D_k \; . \tag{B.41}$$

As n converges to infinity, so does N and, by the above discussion, $x^i(n) - y(n)$ converges to zero, as $n \to \infty$. Consequently, $x^i(n) - x^j(n)$ also converges to zero, for any i,j, completing the proof of part (b).

Finally, since $J(y(n))$ converges and $x^i(n) - y(n)$ converges to zero, part (a) of the theorem follows, as in the proof of Theorem 3.1.■

Proof of Corollary 5.3.1: From part (c) of either Theorem 3.1 or 3.2 and (3.7) we obtain

$$\sum_{i=1}^{M} \sum_{\ell=1}^{L} \sum_{n \in T_\ell^i} \gamma^i(n) g_\ell^i(x^i(n)) < \infty, \qquad \text{a.s.} \tag{B.42}$$

Because of our assumption on the sets T_ℓ^i, it follows that there exists a positive integer c such that, for any i,ℓ,m, the interval $\{cm+1, cm+2, \ldots, c(m+1)\}$ contains at least one element of T_ℓ^i. Let us choose sequences of such elements denoted by $t_{\ell,m}^i$. By (B.42), we have

$$\sum_{i=1}^{M} \sum_{\ell=1}^{L} \sum_{m=1}^{\infty} \gamma^i(t_{\ell,m}^i) g_\ell^i(x^i(t_{\ell,m}^i)) < \infty, \qquad \text{a.s} \tag{B.43}$$

Now notice that, for some constant K_5 0,

$$\gamma^i(t_{\ell,m}^i) \geq \frac{K_4}{t_{\ell,m}^i} \geq \frac{K_4}{c(m+1)} \geq \frac{K_5}{m}, \qquad \forall i,\ell,m\ . \tag{B.44}$$

Hence, (B.43) yields

$$\sum_{m=1}^{\infty} \frac{1}{m} \sum_{i=1}^{M} \sum_{\ell=1}^{L} g_\ell^i(x^i(t_{\ell,m}^i)) < \infty, \qquad \text{a.s.} \tag{B.45}$$

From either Theorem 3.1 or 3.2 and its proof we obtain $\lim_{n\to\infty} (x^i(n)-y(n)) = \lim_{n\to\infty} (y(n+1)-y(n)) = 0$ which implies that

$$\lim_{m\to\infty} (x^i(t_{\ell,m}^i) - y(t_{1,m}^1)) = 0, \qquad \forall i,\ell\ . \tag{B.46}$$

Since J has compact level sets and $J(y(n))$ converges, the sequence $\{y(n)\}$ is bounded. We therefore need to consider the functions g_ℓ^i only on a compact set on which they are uniformly continuous. Therefore,

$$\lim_{m\to\infty} (g_\ell^i(x^i(t_{\ell,m}^i)) - g_\ell^i(y(t_{1,m}^1))) = 0, \qquad \forall i,\ell\ . \tag{B.47}$$

By combining (B.45) and (B.47) we obtain

$$\liminf_{m\to\infty} \sum_{i=1}^{M} \sum_{\ell=1}^{L} g_\ell^i(y(t_{1,m}^1))=0 \ . \tag{B.48}$$

a) By (B.48), there must be some subsequence of $\{t_{1,m}^1\}$ along which $g(y(t_{1,m}^1))$ converges to zero. Let y* be a limit point of the corresponding subsequence of $\{y(t_{1,m}^1)\}$. By continuity, $g(y^*)=0$ and by assumption, y* must be a stationary point of J, so $\nabla J(y^*)=0$. Moreover, $x^i(t_{1,m}^1)$ also converges to y* along the same subsequence. By continuity of ∇J, (3.8) follows.

b) In this case, (B.43) implies

$$\lim_{m\to\infty} \sum_{i=1}^{M} \sum_{\ell=1}^{L} g_\ell^i(x^i(t_{\ell,m}^i))=0, \qquad \text{a.s.} \tag{B.49}$$

and the rest of the proof is the same as for part (a), except that we do not need to restrict ourselves to a convergent subsequence.

c) From part (a) we conclude that some subsequence of $\{y(t_{1,m}^1)\}$ converges to some y* for which $g(y^*)=0$. Consequently, y* minimizes J. Using the continuity of J,

$$\liminf_{n\to\infty} J(y(n)) \le \liminf_{n\to\infty} J(y(t_{1,n}^1)) \le J(y^*) = \inf_{x\in H} J(x) \ .$$

On the other hand, J(y(n)) converges (part (a) of either Theorem 3.1 or 3.2) which shows that (3.10) holds.∎