An Efficient Termination Method for Asynchronous Iterative Algorithms on Message Passing Architectures

Didier EL BAZ

LAAS du CNRS, 7, avenue du Colonel Roche,

31077 Toulouse Cedex, France

e-mail: elbaz@laas.fr

fax number: (33) 61 33 69 69

Abstract : An efficient method for the termination of asynchronous iterative algorithms on message passing architectures is presented. The proposed method is original, general, and requires few control messages. A formal proof of validity of the termination method is given.

Keywords : distributed termination, asynchronous iterative algorithms, systems of equations, Markov systems, fixed point problems, message passing architectures.

1. Introduction

In this paper, we study asynchronous iterative methods for the solution of systems of equations whereby computations are carried out by several processors without any order nor synchronization. We concentrate on the termination of asynchronous iterations in message passing architectures; this problem presents many difficulties since processors possess only local information, there is no global clock, and the communication delays may be arbitrarily large. This topic is related to numerical analysis (termination must occur when the iteration vector is sufficiently close to a solution of the numerical problem) and computer science (a special procedure must be designed for the observation of the processor states, the detection of the termination, and the termination of all the processes). We note that this problem is particularly relevant to the termination detection of distributed processes (see [HaZ87], [Mat87], and [Top84]); however asynchronous iterative algorithms iterate usually indefinitely and computation processes are never idle.

The purpose of this paper is to present an original and efficient method for the termination of asynchronous iterations in message passing systems. The proposed method is general and very economical; more precisely it uses a limited number of control messages and requires less communications than the method proposed by Bertsekas and Tsitsiklis (see [BeT91] and [BeT89]). In particular it does not necessitate the use of acknowledgments of messages. We illustrate asynchronous iterations and the termination method: we consider the solution of Markov systems.

The second Section deals with a brief review of termination methods for asynchronous iterative algorithms. A new termination method is proposed in Section three.

2. The termination problem

In this paper, we consider systems of equations $G(p) = z$ and associated fixed point problems $p = F(p)$, where p, z are vectors of R^n and G, F are mappings from R^n onto itself. Asynchronous iterative algorithms for the solution of these problems are sequences $\{p(k)\}$ of vectors of R^n that can be defined as follows (see [BeT89, Section 6.1]).

2.1. Asynchronous iterations

Definition 2.1. We assume that there is a set of times $T = \{0, 1, 2, ...\}$ at which one or more components of vector p are updated by some processor. Let T^i be the subset of times at which component p_i is updated. For all $i \in \{1, ..., n\}$, the sequences $\{p_i(k)\}$ are defined recursively by:

$$p_i(k+1) = F_i(p_1(\tau_1^i(k)), ..., p_n(\tau_n^i(k))), \forall k \in T^i, p_i(k+1) = p_i(k), \forall k \notin T^i, \tag{2.1}$$

where F_i is the i-th component of the mapping F and

(a) the set T^i is infinite;

(b) $0 \leq \tau_j^i(k) \leq k, \forall j \in \{1, ..., n\}, \forall k \in T^i$;

(c) $\tau_j^i(k)$ is monotonically increasing; $\forall j \in \{1, ..., n\}$,

(d) if $\{k_t\}$ is a sequence of elements of T^i that tends to infinity, then $\lim_{t \to \infty} \tau_j^i(k_t) = +\infty$ for every j.

For further details about asynchronous iterative algorithms the reader is referred to [Bau78], [BeT89], [ChM69], and [Mie75a]. In particular, the convergence of asynchronous iterative algorithms has been established for many problems (see [BeE87], [Elb90], [Elb94a], [EvD91], [Elb96b], [LuM86], [Mie75b], [Mie86], and [MiS85]). Particular attention must be paid to the Asynchronous Convergence Theorem of Bertsekas (see [Ber83] and [BeT89, p. 431]). This theorem is an original and general result, it is also a powerful aid in showing convergence of asynchronous iterative algorithms. The general conditions of the Asynchronous Convergence Theorem of Bertsekas are given as follows:

Assumption 2.1. There is a sequence of nonempty level sets $\{P(k)\}$ with $P(k+1) \subset P(k) \subset R^n, k = 0, 1, ...$ satisfying the following two conditions:

Box Condition: for every k, there exist sets $P_i(k), i = 1, ..., n$, such that $P(k)$ is the Cartesian product: $P(k) = P_1(k) \times ... \times P_n(k)$.

Synchronous Convergence Condition: $F(p) \in P(k+1)$, $\forall k$ and $p \in P(k)$, furthermore if $\{p'(k)\}$ is a sequence such that $p'(k) \in P(k)$ for every k, then every limit point of $\{p'(k)\}$ is a fixed point of F.

It follows from the Asynchronous Convergence Theorem of Bertsekas that if the above conditions are satisfied and the initial solution estimate $p(0)$, belongs to the set $P(0)$, then every limit point of the asynchronous iteration $\{p(k)\}$ is a fixed point of F. Moreover, if the mapping F is monotone increasing, then it follows from assumption (c) that $\{p(k)\}$ converges monotonically from a subsolution or a supersolution (see [Elb90], [Elb94a], [Mie75b], and [Mie86]). However, we note that assumption (c) can be dropped (see for example [Ber83] and [BeT89]).

2.2. Termination procedures for asynchronous iterative algorithms

Termination schemes often used in practice rely on the definition of a local termination condition (see [BeT89, p. 571]). Some schemes consist in the simultaneous observation of the local termination conditions by a special processor. Termination occurs when all local termination conditions are satisfied. In other schemes (see [GiS91]), a special processor collects local termination messages and restart messages sent by the processors. In a different approach (see [MiE89]), the termination scheme samples periodically the process states and deduces the global state. This scheme is based on the communication of Boolean values relative to the satisfaction of a local termination criterion and Boolean fixed point computation via an asynchronous iterative method. The solution of the Boolean fixed point problem gives the global state. However, each processor must possess estimates of the initialization and termination times of the Boolean asynchronous iterative method. Another scheme (see [ChZ91]) uses termination messages and acknowledgment of termination messages. A processor terminates if its local termination condition is satisfied and it has received termination messages and acknowledgments of its termination messages from all other processors. In a message passing system, no formal proof of validity exists for the above termination schemes since processors do not have access to a global clock and messages may be subject to unknown delays. However, the above termination schemes present a real practical interest.

An interesting approach to the termination problem has been proposed by Bertsekas and Tsitsiklis (see [BeT89, Section 8] and [BeT91]). The problem is decomposed into two parts. First the asynchronous iterative algorithm is modified so that it converges to a fixed point sufficiently close to the solution and terminates in finite time. Then a termination detection procedure is applied. Bertsekas and Tsitsiklis propose the following modification of the asynchronous iterative algorithm: if the update of a component of the iteration vector is not significantly different from the previous value of this component, then the update is not taken into account nor it is communicated to other processors. In this case, termination of the algorithm occurs when an update causes no change in the value of the components for all processors (i.e. all local termination conditions are satisfied) and no message is in transit. Several procedures can be used for the detection of the termination. We can quote for example the procedure of Dijkstra and Scholten (see [DiS80], see also [BeT89]) which is based on message acknowledgments and generation of an activity graph, and the snapshot algorithm of Chandy and Lamport (see [ChL85], see also [BeT89]). The latter approach is based on marker messages generation and processor and link states recording at delivery of marker messages. The states recorded in a snapshot do not correspond to a true global state of the computing system at a given time. However, the information contained in a snapshot is sufficient for the detection of certain properties of the true global state of the computing system and in particular: termination. To the best of our knowledges the method of Bertsekas and Tsitsiklis is one of the rare methods in the literature for which there exists a rigorous proof of validity. However, the method presents several drawbacks: it requires the use of a complex protocol and more message exchanges than the original asynchronous iterative algorithm defined by (2.1). Finally, hypotheses which are stronger than the conditions of the Asynchronous Convergence Theorem of Bertsekas must be satisfied so that the modified asynchronous iterative algorithm converges in finite time.

Another very interesting approach has been proposed by Savari and Bertsekas (see [SaB94]). The main feature of this approach is that asynchronous iterative algorithms are slightly modified: every new update is taken into account and each processor sends an update if and only if the value of the component is modified. Requests are also broadcasted in the whole network as often as a local termination condition is not satisfied. A processor is active as long as its local termination condition is unsatisfied or requests are received. Termination occurs when all processors are idle and no updates nor requests are in transit. Termination is detected using a standard termination detection protocol (see [ChL85], [DiS80], and [Top84]). Savari and Bertsekas have given a formal proof of validity of this termination method. The main advantage of this method is that it can be applied with success to a wider class of problems and algorithms than the method of Bertsekas and Tsitsiklis; its main drawback is that a very large number of communications is required. Acknowledgments are sent to adjacent processors in the former method (see [BeT89]), whereas requests must be broadcasted in the whole network in the latter method.

Finally, we quote an approach of the author (see [ElB96a]) which is different from the approach presented in this paper: it is based on the intensive use of level sets and the Asynchronous Convergence Theorem of Bertsekas. The termination method requires slight modifications of asynchronous iterations and few extra communications. Termination occurs when the iteration vector belongs to a given level set $P(\hat{q})$. The index of the smallest level subset to which each component of the iteration vector belongs is computed at each updating; this index is sent together with the update. If this index is greater than or equal to $\hat{q}$, then the component is abandoned. We note in particular that the computation of the indices of level sets is not time consuming and only one integer is added to each message. The resulting method is simple and original.

3. A new termination method

In this Section we present an original termination procedure. We illustrate asynchronous iterations and the proposed termination procedure on an example: the solution of Markov systems via Richardson's method. The reader is referred in particular to [LuM86], [BLA87], [BeT89], and [Elb94b] for previous contributions to this topic.

3.1. Numerical problem and iterative algorithm

We consider the system:

$$Qp = 0, \tag{3.1}$$

where p is a stationary probability vector of dimension n ($\sum_{i=1}^{n} p_i = 1$) and Q the so-called transition matrix is an $n \times n$ irreducible aperiodic Z-matrix ($q_{ii} > 0, \forall i; q_{ij} \leq 0, \forall i, j$, with $j \neq i$) such that $\sum_{k=1}^{n} q_{ki} = 0, \forall i$, moreover the rank of matrix Q is equal to $n-1$. Under the above assumptions, there exists a unique vector p^* which is the solution of (3.1) such that $\sum_{i=1}^{n} p_i^* = 1$, and $p^* > 0$ (see [BeT89]). Markov systems occur in many practical situations, we can quote for example circuit-switched networks modelling (see [BLA87]).

Contrarily to the approaches presented in [BLA87] and [LuM86], where all the components of p^* are computed via an iterative method, we constrain a component of vector p, say p_n to be 1 and consider the new system:

$$\sum_{j=1}^{n-1} q_{ij}.p_j + q_{in} = 0, i = 1, ..., n-1. \tag{3.2}$$

From the solution $\hat{p}$ of system (3.2) we can obtain the solution p^* of system (3.1), we have:

$$p_i^* = \frac{\hat{p}_i}{\sum_{i=1}^{n} \hat{p}_i}, i = 1, ..., n.$$

We now introduce the probability vector $p(0)$ with components:

$$p_1(0) =, ..., = p_{n-1}(0) = 0, p_n(0) = 1.$$

Since $q_{in} \leq 0, i = 1, ..., n-1$, we have:

$$\sum_{j=1}^{n-1} q_{ij}.p_j(0) + q_{in} \leq 0, i = 1, ..., n-1.$$

In order to simplify presentation, p will denote in the sequel the vector of R^{n-1} with components $p_i, i = 1, ..., n-1$, since p_n is a constant. We consider the solution of system (3.2) via Richardson's method. The associated fixed point mapping $F : R^{n-1} \rightarrow R^{n-1}$, is defined by

$$F_i(p) = p_i - \alpha.(\sum_{j=1}^{n-1} q_{ij}.p_j + q_{in}), i = 1, ..., n-1,$$

where α is a positive constant. Clearly Richardson's method is well suited to parallel or distributed computation: the components of the iteration vector can be updated independently by different processors. Moreover we can consider an asynchronous implementation. El Baz has proven the monotone convergence of asynchronous Richardson's algorithms from $p(0) = 0$, in [Elb94b]. This unreported property of Markov systems follows from general convergence results presented by the author in [Elb90] and [Elb94a], the monotone convergence follows in particular from assumption (c) of Section 2, the continuity, and the monotone increasing property of the mapping F (see [Elb90], [Elb94a], [Elb94b], and [Mie75b]). It can easily be shown that the conditions of the Asynchronous Convergence Theorem of Bertsekas are also satisfied in this case.

3.2. Termination procedure

In this subsection we present a new termination method for asynchronous iterations. As in [BeT91], asynchronous iterations are modified in order to terminate in finite time. However, the termination procedure requires no message acknowledgments. So, there are less message exchanges than in the method proposed in [BeT91]. The termination procedure is well suited to the application considered in this Section. We note that this procedure is not limited to Markov chains; it can be used for many problems like network flow and fixed point problems.

In order to simplify presentation we will assume that communications are asynchronous. However, if communications are synchronous, then a simple solution consists in the implementation of an additional buffer process; for more details the reader is referred to [Elb93]. We make the following assumption on message transmission:

Assumption 3.1. The transmission time t_t of messages between any pair of adjacent processors is bounded by a positive constant δ, we have:

$$t_t < \delta.$$

The modified asynchronous iterative algorithm is represented by the finite state machine given in figure 1. A processor has three possible states: active (A), idle (I), and terminal (T). Initially, only one processor is active namely the root processor which is denoted by R and all other processors are idle.

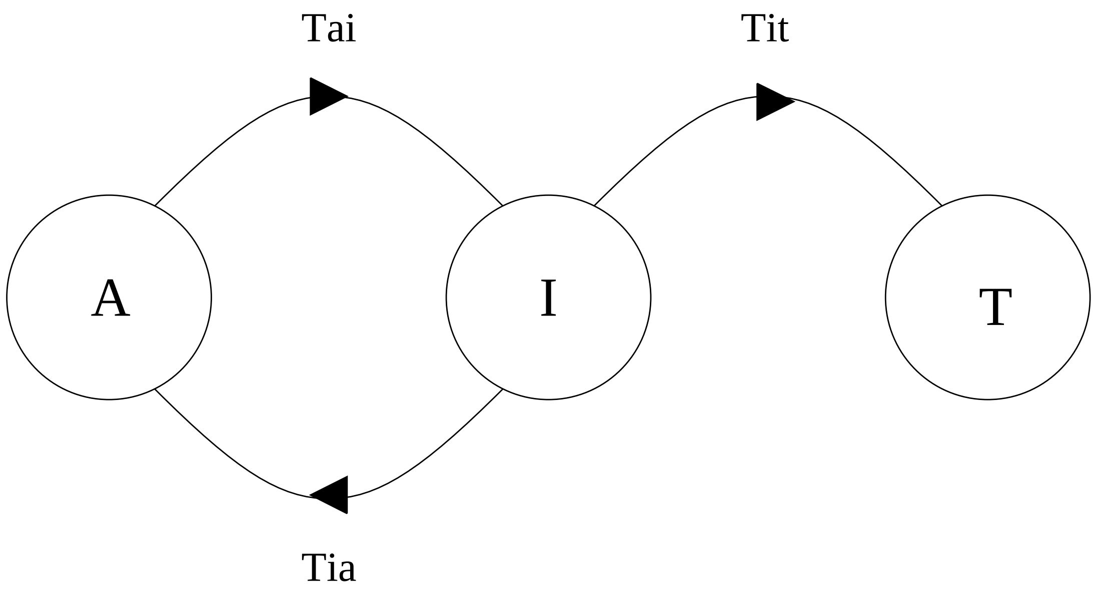

Figure 1: finite state machine

Four types of messages are issued from each processor: updates, active messages, idle messages, and termination messages. The following data are stored in each processor P :

- the identity of the processor who has activated P, namely, the father of P;

- the list of processors activated by P, namely, the sons of P;

- the counter of updates which have been received by P;

- the updates.

State A: In the active state, a processor P resets the counter of received updates and then computes the local termination test on the basis of the most recent updates which are available in the buffers. If the local termination test is satisfied, then no updating is made, else the components of the iteration vector assigned to processor P are updated, updates are sent to adjacent processors, and the processor P waits for active messages during $2.\delta$. A processor P adds P' to the list of its sons, upon receiving an active message from a processor P'. A processor P withdraws P' from the list of its sons, upon receiving an an idle message from a processor P'. For the above application, we can use the following local termination test: for all i element of $I(P)$, the subset of indices of the components of the vector p which are updated by the processor P,

$$\max_{i \in I(P)} \left| \sum_{j=1}^{n-1} q_{ij}.p_j + q_{in} \right| \leq \epsilon, \tag{3.3}$$

where p_j is the most recent update of the j-th component of vector p available in the buffer of P and ϵ is a given positive constant. In general this test is not time consuming.

State I: In the idle state, a processor is waiting for messages.

State T: In the terminal state, a processor does nothing.

Transition Tai: An active processor P changes its state to idle if and only if the local termination test is satisfied, the list of its sons is empty, and the counter of received updates is equal to zero (i.e. no new update has been received). The processor P sends an idle message to its father when the transition Tai is fired.

Transition Tia: In the idle state, a processor P changes to the active state upon receiving a new update. When the transition Tia is fired, the processor P sends an active message to its father (i.e. to the processor which has sent an update to P) and resets the identity of its father.

Transition Tit: In the idle state, the root processor R changes immediately to the terminal state and termination messages are sent to adjacent nodes in a given spanning tree T of the network of processors, which is rooted at R. An idle processor P different from R changes to the terminal state upon receiving a termination message from the lower level adjacent node in the tree T. When the transition Tit is fired, the processor P sends a termination message to upper level adjacent nodes in the tree T.

The method working can be summarized as follows: at the beginning the root R is the only active processor and all other processors are idle. All processors become progressively active upon receiving updates from other processors. An activity graph is generated. The activity graph moves on as the computation progresses, updates are received, and local termination tests are satisfied.

Now we present conditions under which the modified asynchronous iterative algorithm converges in finite time. Next we will give a rigorous proof of validity of the above termination procedure.

Remark 3.1. In the modified asynchronous iterative algorithm proposed above, some components of the iteration vector eventually become constant after a finite time. Thus, the convergence of the modified asynchronous iterative algorithm to a limit point does not follow directly from the general Asynchronous Convergence Theorem of Bertsekas. However, among the different classes of problems for which the conditions of the Asynchronous Convergence Theorem are satisfied, we can identify two large subclasses for which the modified asynchronous iterative algorithm converges in finite time (see in particular [BeT91]):

(1) the case where the associated fixed point mapping F satisfies a weighted maximum norm contraction condition:

$$\|F(p) - F(p')\| \leq \alpha. \|p - p'\|, \forall p, p' \in R^n,$$

where $\alpha \in [0, 1)$ and $\|p\| = \max_i \frac{|p_i|}{w_i}$ is a weighted maximum norm (see [BeT89, p 434]).

(2) the case where the mapping Q considered for example in the system (3.1) is off-diagonally antitone, strictly diagonally isotone, and there exists a subsolution p and a supersolution p' such that $p \leq p'$ and $Qp \leq 0 \leq Qp'$. This case occurs in many practical situations like for example: Markov systems, network flow, and boundary value problems. It follows from Assumption 3.1 that the modified asynchronous iterative algorithms converge monotonically from a supersolution or a subsolution, respectively to a limit point which is a supersolution or a subsolution, respectively (see [Elb90], [Elb94b], and [Mie75b]).

Remark 3.2. If a processor P is not active on and after a time t; then it follows from the definition of State A and Assumption 3.1 that all the neighbors of P will have the final value of the components of the iteration vector which are updated by P, after the time $t + \delta$.

Lemma 3.1. If a processor P is active, then its father P' is active.

Proof: It follows from the definition of state A that every active processor P' waits for active messages during $2.\delta$, after the emission of an update at time t. If a neighbor P of a processor P' is idle, then it follows from Assumption 3.1 and the definition of transition Tia that P becomes active and sends an active message to P' before the time $t + \delta$. Moreover it follows from Assumption 3.1 that the processor P' will receive the active message before the time $t + 2.\delta$. By the definition of state A, the processor P' adds P to the list of its sons upon receiving the active message from P and it follows from the definition of transition Tai that P' can change its state to idle only if P sends an idle message to P'. □

Corollary 3.1. The activity graph is a tree of directed arcs that is rooted at R.

Proof: Initially, the activity graph contains only the root R, thus it is trivially a tree. Assume now that the activity graph is a directed tree rooted at R, before any firing of a transition. If we assume that a transition Tai is fired, then it follows from Lemma 3.1 that only a processor P without son can fire a Transition Tai. Hence, the node P and the directed arc (P', P) are withdrawed from the activity tree, P' being the unique father of P, so the resulting activity graph is still a directed tree rooted at R. Assume now that a processor P fires a transition Tia, then the node P and the directed arc (P', P) are added to the activity tree, P' being the unique father of P. Moreover by the definition of transition Tia, P cannot be in any cycle since it has a degree 1 in the activity graph, so the resulting activity graph is still a directed tree rooted at R. Finally, it follows from the definition of state A and I that all processors will never be active on and after the time at which R is withdrawed from the activity tree. □

Corollary 3.2. Assume that R is idle. Then the value of the components of the iteration vector available in the buffer of each processor corresponds to the last update and all local termination conditions are satisfied.

Proof: The result follows from the definition of transition Tai, Corollary 3.1, and Remark 3.2. □

Lemma 3.2. Assume that all local termination conditions are satisfied on and after a time t. Then all processors will be idle in finite time after t.

Proof: If all local termination conditions are satisfied on and after a time t, then it follows from the definition of State A that no updates are sent after time t. Moreover by using Assumption 3.1, the definition of transition Tai, Lemma 3.1, and Corollary 3.1, it can be shown by a simple induction over the activity tree topology that all processors will become idle in finite time after t. □

Lemma 3.3. If all the processors are idle at time t, then all the processors will terminate in finite time after t.

Proof: This result follows from the definition of transition Tit and Assumption 3.1 by a simple induction over the topology of tree T. □

Proposition 3.1. Assume that we are in the case (1) or (2) of Remark 3.1. Then the termination of all the processors will occur in finite time.

Proof: It follows from Remark 3.1 that the modified asynchronous iterative algorithm converges in finite time. Hence, it follows from Lemmas 3.2 and 3.3 that all the processors terminate in finite time. □

Proposition 3.2. If a processor has terminated, then the global termination condition is satisfied.

Proof: By an induction over the topology of the tree T using the definition of transition Tit, we can show that if a processor has terminated, then R has also terminated and it follows from the definition of transition Tit and Corollary 3.2 that the value of the components of the iteration vector available in the buffer of each processor is relative to the last update and all local termination conditions are satisfied. So the global termination condition which is a combination of all local termination conditions is also satisfied. □

References

[Bau78] Baudet, G.M. 'Asynchronous iterative methods for multiprocessors', *J. Assoc. Comput. Mach.*, 2, pp. 226-244, 1978.

[BLA87] Bernussou, J. Le Gall, F. and Authie, G. 'About some iterative synchronous and asynchronous methods for Markov chain distribution computation', *Proceedings of 10-th IFAC World Congress*, 1987.

[Ber83] Bertsekas, D.P. 'Distributed asynchronous computation of fixed points', *Mathematical Programming*, 27, pp. 107-120, 1983.

[BeE87] Bertsekas, D.P. and El Baz, D. 'Distributed asynchronous relaxation methods for convex network flow problems', *SIAM J. on Control and Optimization*, 25, pp. 74-85, 1987.

[BeT89] Bertsekas, D.P. and Tsitsiklis, J. *Parallel and Distributed Computation, Numerical Methods*, Prentice Hall, Englewood Cliffs, New Jersey, 1989.

[BeT91] Bertsekas, D.P. and Tsitsiklis, J. 'Some aspects of parallel and distributed iterative algorithms. A survey', *Automatica,* vol 27, 1, pp. 3-21, 1991.

[ChL85] Chandy, K.M., and Lamport L. 'Distributed snapshots: Determining global state of distributed systems', *ACM Trans. on Computer Systems,* 3, pp. 63-75, 1985.

[ChM69] Chazan, D. and Miranker, W. 'Chaotic relaxation', *Linear Algebra Appl.*, 2, pp. 199-222, 1969.

[ChZ91] Chajakis, E. and Zenios, S.A. 'Synchronous and asynchronous implementations of relaxation algorithms for nonlinear network optimization', *Parallel Computing*, 17, pp. 873-894, 1991.

[DiS80] Dijkstra, E.W., and Scholten, C.S. 'Termination detection for diffusing computation', *Information Processsing Letters'*, 11, pp. 1-4, 1980.

[Elb90] El Baz, D. 'M-functions, and parallel asynchronous algorithms', *SIAM Journal on Numerical Analysis*, 27, pp. 136-140, 1990.

[Elb93] El Baz, D. 'Asynchronous implementation of relaxation and gradient algorithms for convex network flow problems', *Parallel Computing*, 19, pp. 1019-1028, 1993.

[Elb94a] El Baz, D. 'Nonlinear systems of equations and asynchronous iterative algorithms', *Advances in Parallel Computing*, 9, pp. 89-96, 1994.

[Elb94b] El Baz, D. 'Parallel iterative algorithms for the solution of Markov systems', *Proceedings of the 33 rd IEEE Conference on Decision and Control*, Orlando, USA, pp. 2524-2527, 1994.

[Elb96a] El Baz, D. 'A method for terminating asynchronous iterative algorithms on message passing systems, *Parallel Algorithms and Applications* 11, 2, 1996.

[Elb96b] El Baz, D. 'Asynchronous gradient algorithms for a class of convex separable network flow problems', to appear in *Computational Optimization and Applications*, 5, pp. 187-205, 1996.

[EvD91] Evans, D.J. and Deren, W. 'An asynchronous parallel algorithm for solving a class of nonlinear simultaneous equations', *Parallel Computing*, 17, pp. 165-180, 1991.

[GiS91] Giraud, L. and Spiteri, P. 'Résolution parallèle de problèmes aux limites non linéaires', *MMAN,* 25, pp. 579-606, 1991.

[HaZ87] Hazary C. and Zedan H. ' A distributed algorithm for distributed termination', *Information Processing Letters*, 24, pp. 293-297, 1987.

[LuM86] Lubachevsky, B. and Mitra, D. 'A chaotic asynchronous algorithm for computing the fixed point of a nonnegative matrix of unit spectral radius', *J.A.C.M.*, 33, pp. 130-150, 1986.

[Mat87] Mattern F. 'Algorithms for distributed termination detection', *Distributed Computation*, 2, pp. 161-175, 1987.

[Mie75a] Miellou, J.C. 'Algorithmes de relaxation chaotique à retards', *RAIRO*, R1, pp. 55-82, 1975.

[Mie75b] Miellou, J.C. 'Itérations chaotiques à retards, étude de la convergence dans le cas d'espaces partiellement ordonnés', *C.R.A.S. Paris*, 280, pp. 233-236, 1975.

[Mie86] Miellou, J. C. 'Asynchronous iterations and order intervals', in *Parallel Algorithms and Architectures*, M. Cosnard ed., North Holland, pp. 85-96, 1986.

[MiE89] Miellou, J.C. and Evans D.J. 'Stopping criteria for parallel asynchronous algorithms', *Computer Studies 503, Loughborough Institute of Technology*, November 1989.

[MiS85] J. C. Miellou, P. Spiteri, 'Un critère de convergence pour des méthodes générales de point fixe $'$, *R.A.I.R.O. MMAN*, 19 (1985), pp. 645-669.

[SaB94] Savari, S.A. and Bertsekas D.P. 'Finite termination of asynchronous iterative algorithms', *LIDS report P-2245, M.I.T.*, April 1994.

[Top84] Topor, R.W. 'Termination detection for distributed computation', *Inform. Process. Lett.*, vol 18, pp. 33-36, 1984.