Parallel Iterative Algorithms for the Solution of Markov Systems

Didier El Baz

LAAS du CNRS, 7, avenue du Colonel Roche, 31077 Toulouse Cedex France

Abstract

We propose various parallel synchronous and asynchronous iterative algorithms for the solution of Markov systems. We comment on the convergence of these algorithms. We present and analyze a computational experience using a distributed memory multiprocessor.

1. Introduction

Several authors have proposed parallel iterative algorithms for the solution of Markov systems (see [2], [4], [10], and [11]). In particular asynchronous iterative algorithms have been presented in [2], [4], and [11]. In this study we concentrate on the solution of Markov systems via parallel versions of relaxation and Richardson's methods. We comment on the monotone convergence of asynchronous and synchronous algorithms. This property is unreported in the case of Markov systems and follows from general convergence results presented by the author in [6] and [8]. Finally, computational results on a message passing architecture are turned out and analyzed.

The second Section deals with the solution of Markov systems via parallel iterative algorithms. Experimental results are presented in Section three.

2. Problem and Parallel Algorithms

We consider Markov systems:

$$Qp = 0, \tag{2.1}$$

where p is a stationary probability vector of dimension n ($\sum_{i=1}^{n} p_i = 1$), Q the so-called transition matrix is an $n \times n$ irreducible aperiodic Z-matrix (i.e. $q_{ii} > 0, \forall i; q_{ij} \leq 0, \forall i, j$, with $j \neq i$) such that $\sum_{k=1}^{n} q_{ki} = 0, \forall i$. Then, there exists a unique vector p^* solution of (2.1) and such that $\sum_{i=1}^{n} p_i^* = 1$, furthermore $p^* > 0$ (see [4]). Markov systems occur in many practical situations, we can quote for example circuit-switched networks modelling (see [2] and [10]).

Contrarily to the approaches presented in [2], [10], and [11] where all the components of the iterate vector solution of (2.1) are computed via an iterative method, we fix here a component of vector p say p_n to 1 and delete the last equation in (2.1), in the sequel we will draw an inference from this choice with respect to the convergence properties of the proposed iterative methods. We have the new system:

$$\sum_{j=1}^{n-1} q_{ij}.p_j + q_{in} = 0, i = 1, ..., n-1. \tag{2.2}$$

From the solution $\hat{p} > 0$ of system (2.2) we can obtain the probability vector p^* solution of system (2.1):

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

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

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

Since $q_{in} \leq 0, i = 1, ..., n-1, p(0)$ satisfies:

$$\sum_{j=1}^{n-1} q_{ij}.p_j(0) + q_{in} \leq 0, i = 1, ..., n-1. \tag{2.3}$$

For simplicity of 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 define the set $P = \{p \in R^{n-1} / p(0) \leq p \leq \hat{p}\}$.

We consider the solution of system (2.2) via iterative methods and more particularly via relaxation and Richardson's method. Associated fixed point mappings can be defined: the relaxation mapping $F : P \subset R^{n-1} \to R^{n-1}$ with components $F_i(p) = \tilde{p}_i$ such that

$$\sum_{j=1, j \neq i}^{n-1} q_{ij}.p_j + q_{ii}.\tilde{p}_i + q_{in} = 0, i = 1, ..., n-1;$$

Richardson's mapping $F' : P \subset R^{n-1} \to R^{n-1}$ with components

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

where α is a positive constant. Clearly, relaxation and Richardson's method are well suited to parallel computation: components of the iterate vector can be updated independently by different processors. We can consider synchronous algorithms whereby computations are carried out according to a particular order with synchronization points. We can also consider more general procedures whereby computations are carried out without any order neither synchronization, namely asynchronous algorithms. Briefly, an asynchronous iterative algorithm associated with the fixed point problem $p = F(p)$ is a sequence $\{p(k)\}$ of vectors of R^{n-1} which satisfies the following general definition (see [3]- [4, Section 6.1]). 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 $i = 1, ..., n-1$, the sequences $p_i(k)$ are defined recursively by:

$$p_i(k+1) = F_i(p_1(\tau_1^i(k)), ..., p_{n-1}(\tau_{n-1}^i(k))), k \in T^i \tag{2.4}$$

$$p_i(k+1) = p_i(k), k \notin T^i,$$

where for $i = 1, ..., n-1$:
(a) the set T^i is infinite,
(b) $0 \leq \tau_j^i(k) \leq k$, $j = 1, ..., n-1$, $\forall k \in T^i$,
(c) $\tau_j^i(k)$ is monotone increasing, $j = 1, ..., n-1$,
(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 [1], [4], [5], and [12]. Convergence of asynchronous iterative algorithms has been established for many problems (see [1], [3]-[6], [8], and [11]-[12]). Particular attention must be paid to the Asynchronous Convergence Theorem of Bertsekas (see [3] and [4, p. 431]). This theorem is an original and general result, it is also a powerful aid in showing convergence of asynchronous iterative algorithms.

More particularly, if the mapping F is continuous and isotone (i.e. monotone increasing) then, we can show using (c) that $\{p(k)\}$ converges monotonically to a solution of the fixed point problem from a subsolution or a supersolution (see [6], [8], and [12]). We note that assumption (c) is not always introduced to show the convergence of asynchronous iterations (see for example [3] and [4]).

Proposition 2.1: Asynchronous and synchronous relaxation and Richardson's methods converge monotonically to $\hat{p}$ from $p(0)$.

Proof: (I) From the definitions of $F, p(0)$, and $\hat{p}$, it follows that the relaxation mapping F is well defined, continuous, and isotone on P since Q is a Z-matrix (see [6, Lemmas 2.1 and 2.2] see also [8, Proposition 2.1]). From (2.3) we can show that $p(0) \leq F(p(0))$ (see [6, Lemma 2.2]), so $p(0)$ is a subsolution. Hence, using assumption (c) we can show the monotone convergence of asynchronous and synchronous relaxation algorithms starting from $p(0)$ (see [6, Theorem 3.1]).

(II) We note that the linear operator associated with matrix Q satifies clearly the Lipschitz Continuity Assumption 2.3 of [8] on P, i.e. there exists a constant β such that

$$\|Q(p - p')\|_2 \leq \beta. \|p - p'\|_2, \forall p, p' \in P.$$

One can take merely β greater or equal to the matrix norm $\|Q\|_2$. Hence, by setting $\alpha = \frac{1}{\beta}$ and using the definition of mapping F' and the fact that Q is a Z-matrix, we conclude that F' is continuous and isotone on P (see [8, Proposition 2.3]). Moreover it follows from (2.3) and the definition of F' that $p(0) \leq F'(p(0))$. Then using assumption (c) we can show the monotone convergence of asynchronous and synchronous Richardson's algorithms starting from $p(0)$ (see [8, Proposition 2.4]). In practice one can take α greater than $\frac{1}{\|Q\|_2}$ since we consider only the interval $P = \{p \in R^{n-1} / p(0) \leq p \leq \hat{p}\}$ instead of R^{n-1}.

3. Computational Experience

Computational experiments are carried out using the transputer based distributed memory multiprocessor Tnode 16-32. Richardson's and relaxation methods are implemented using one processor, parallel relaxation and Richardson's algorithms are implemented using 2,3,4,5,6,7, and 8 processors. We consider a bloc tridiagonal matrix Q issued from circuit-switched networks modelling and the splitting of the probability vector p into subsets of components. Each subset is relative to a diagonal bloc of matrix Q (e.g. constant number of communications in the network, in the case of circuit-switched networks modelling) and associated with a processor for easiness of communication. The diagonal blocs have different sizes. However we always try to balance as best as we can the number of components updated by the different processors. Scheduling of computational tasks is made according to static mode. We use a pipeline network of processors with bidirectional links. This topology seems naturally well suited for the solution of bloc tridiagonal systems by means of the splitting presented in this paper. A detailed description of the implementation of asynchronous iterative algorithms using the T-node can be found in [7]

We consider a problem with probability vector size close to 100. We take a stepsize $\alpha = 0.08$ for Richardson's method. The starting point is always: $p_i(0) = 0, i = 1, ..., n-1$.

We use a termination method designed for pipeline processor topology and issued from the algorithm proposed in [4, subsection 8.1], a formal proof of validity of the termination method is given in [9]. In particular for all processors P_r, the local termination condition is: $\left|\sum_{j=1}^{n-1} q_{ij}.p_j + q_{in}\right| \leq 0.001, \forall i \in I(P_r)$, where $I(P_r)$ is the set of indices of the components of the vector p updated by P_r, and p_j are the values of the components of p available in the buffers of P_r.

Tables 1 to 4 give the number of iterations or minimum and maximum numbers of iterations, solution time in milliseconds, speedup and efficiency of the different iterative methods. Tables 1 and 2 point out that asynchronous Richardson's methods are generally faster than synchronous Richardson's methods. In the case of 7 processors the loads are particularly well balanced, this fact can explain the good performances of synchronous Richardson. We note that asynchronous implementation speeds up efficiently Richardson's method.

The relaxation method is faster than Richardson's method. However synchronous and asynchronous implementations do not speed up efficiently the relaxation method. This is in part due to the fact that the relaxation method which is a Gauss-Seidel iterative scheme when it is implemented using one processor becomes a Jacobi iterative scheme when it is implemented using several processors.

Asynchronous relaxation algorithms are generally faster than their synchronous counterparts. The speedup of asynchronous algorithms increases generally with the number of processors, the same kind of remark does not hold in the case of synchronous algorithms since loads which are not well balanced cause idle time that may be large. We note that the solution times of asynchronous relaxation and Richardson's methods and synchronous relaxation tend to be close when the number of processors increases. Finally we point out that performances of the parallel iterative algorithms studied in this paper should be better for problems with larger size.

References

1. G.M. Baudet, "Asynchronous iterative methods for multiprocessors," J. Assoc. Comput. Mach., 2, pp. 226-244, 1978.

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

3. D.P. Bertsekas, "Distributed asynchronous computation of fixed points", Mathematical Programming, vol 27, pp. 107-120, 1983.

4. D.P. Bertsekas, and J. Tsitsiklis, Parallel and Distributed Computation, Prentice Hall, Englewood Cliffs, New Jersey, 1989.

5. D. Chazan, and W. Miranker, "Chaotic relaxation," Linear Algebra Appl., 2, pp. 199-222, 1969.

6. D. El Baz, "M-functions, and parallel asynchronous algorithms," SIAM J. on Numer. Anal., 27, pp. 136-140, 1990.

7. D. El Baz, "Asynchronous implementation of relaxation and gradient algorithms for convex network flow problems," Parallel Computing, 19, pp. 1019-1028, 1993.

8. D. El Baz, "Nonlinear systems of equations and asynchronous iterative algorithms," in Advances in Parallel Computing, 9, North Holland, 1994, pp. 89-96.

9. D. El Baz, "A method of terminating parallel asynchronous iterative algorithms on message passing architectures," LAAS Report 94044, february 1994.

processors	1	2	3	4	5	6	7	8
iterations	297	297	297	297	297	297	297	297
time	2108.736	1202.688	842.368	728.512	733.504	536.384	412.928	418.048
speedup	1.0	1.753	2.503	2.895	2.875	3.931	5.107	5.044
efficiency	1.0	0.877	0.834	0.724	0.575	0.655	0.730	0.631

Table 1 : synchronous Richardson

processors	1	2	3	4	5	6	7	8
iterations	297	253-308	255-336	237-457	207-435	235-352	257-338	239-393
time	2108.736	1119.104	812.608	658.368	578.496	468.928	414.144	386.624
speedup	1.0	1.884	2.595	3.203	3.645	4.497	5.092	5.454
efficiency	1.0	0.942	0.865	0.801	0.729	0.749	0.727	0.682

Table 2 : asynchronous Richardson

processors	1	2	3	4	5	6	7	8
iterations	141	170	188	197	222	241	246	250
time	823.616	684.160	527.616	487.04	546.752	431.680	341.504	351.616
speedup	1.0	1.204	1.561	1.691	1.506	1.908	2.412	2.342
efficiency	1.0	0.602	0.52	0.423	0.301	0.318	0.345	0.293

Table 3 : synchronous relaxation

processors	1	2	3	4	5	6	7	8
iterations	141	136-167	169-221	167-261	168-360	191-299	239-309	184-384
time	823.616	574.976	512.576	442.304	450.112	377.856	368.384	347.904
speedup	1.0	1.432	1.607	1.862	1.830	2.180	2.236	2.367
efficiency	1.0	0.716	0.536	0.466	0.366	0.363	0.319	0.296

Table 4 : asynchronous relaxation

10. F. Le Gall, "Solving a bloc-tridiagonal Markov system on Parallel Computers," in Advances in Parallel Computing, 4, North Holland, 1992, pp. 213-220,
11. B. Lubachevsky, and D. Mitra, "A chaotic asynchronous algorithm for computing the fixed point of a nonnegative matrix of unit spectral radius," J.A.C.M., 33, 1, pp. 130-150, 1986.
12. J.C. Miellou, "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.