Journal of Economic Dynamics and Control 12 (1988) 167–172. North-Holland

ASYNCHRONOUS ALGORITHMS IN NON-COOPERATIVE GAMES*

Tamer BAŞAR

University of Illinois at Urbana-Champaign, Urbana, IL 61801, USA

1. Introduction

An equilibrium solution of an N-person game is said to be *globally stable* (or, simply, *stable*) with respect to an *a priori* fixed ordering of computation of policies if, regardless of what initial policies the players start with, the generated algorithm converges to that equilibrium. At each step of such an iteration, each player determines his policy (or action, if the underlying game is static) by optimizing his individual cost function, using the most recently available policies of the other players as given. Clearly, any deviation from such an equilibrium followed by an optimal adjustment process that respects the given fixed ordering for computation of policies, would lead back to the same equilibrium, and furthermore a globally stable equilibrium is necessarily unique [Başar and Olsder (1982), Başar (1986)]. An equilibrium solution is said to be *locally stable*, if it is globally stable in a domain which is an open neighborhood of the equilibrium solution, and naturally a locally stable equilibrium need not be unique.

If the order in which the players respond to a non-equilibrium point and the information available to them at the time of their computation are not predetermined, then the notion of a stable equilibrium becomes more restrictive. Existence of such equilibria can best be studied by treating the underlying algorithm as an *asynchronous algorithm* [Bertsekas (1983), Tsitsiklis et al. (1986), Li and Başar (1987a)] and we will refer to such equilibria as *asynchronously stable equilibria*.

Stable equilibria (be they synchronous or asynchronous) are appealing since they allow for real-time iterative computation of non-cooperative equilibria. The only information carried along the line is the most current policy information (or action information, in the case of static games), and each

*This work was supported in part by the Air Force Office of Scientific Research under Grant No. AFOSR 084-0056. A fuller version of the paper under the title 'Relaxation Techniques and Asynchronous Algorithms for On-Line Computation of Noncooperative Equilibria' can be obtained from the author.

player need not know the others' objective functionals. This appealing feature brings with it some conditions which further restrict the Nash and/or saddle-point equilibria. Therefore, the question naturally arises as to whether the stringent conditions imposed by stable equilibria can be relaxed by adopting a more general class of algorithms which would still retain some of the appealing informational constraints that may be imposed on the problem. The main message of this talk has been that such a relaxation is possible for both zero-sum and non-zero-sum games, and below we outline some of the results in this direction under the asynchronous mode of operation.

2. Stability of equilibria in two-person games

A two-person non-zero-sum (static or dynamic) game in normal form is completely characterized by the policy (strategy) spaces U, V for players 1, 2, respectively, and the objective functions (say, to be minimized) J^1, J^2, the superscript being the identifier for the corresponding player. Let u, v denote the respective policy variables of the two players. Then, one possible iteration that leads to non-cooperative equilibrium solution is

$$\begin{aligned} u_{k+1} &= \arg\min_{u \in U} J^1(u, v_k) := L^1(v_k), \\ v_{k+1} &= \arg\min_{v \in V} J^2(u_{k+1}, v_k) := L^2(u_k), \\ k &= 0, 1, \ldots, \qquad v_0 \in V \quad \text{specified}. \end{aligned} \tag{1}$$

This is akin to the *Gauss–Seidel* procedure, where the players take their turns sequentially and act on the most recent policy information obtained from the other player. Here L^i is the optimum response function of player i $(i = 1, 2)$, and the iteration can equivalently be expressed as

$$v_{k+1} = (L^2 \circ L^1)(v_k) := L(v_k), \qquad v_0 \in V. \tag{2}$$

An equilibrium solution (u^*, v^*) of the game $(J^1, J^2; U, V)$ is *stable* if, and only if, it is the limit point of the sequence $\{u_k, v_k\}_{k=0}^{\infty}$ generated by (2), for all $v_0 \in V$.

A second iteration would be obtained, if we assume that the players compute in parallel, in which case (1) is replaced by

$$u_{k+1} = L^1(v_k), \quad v_{k+1} = L^2(u_k), \qquad u_0 \in U, \quad v_0 \in V. \tag{3}$$

This is related to the *Jacobi* computational procedure and is also known as the *Cournot iteration*. The counterpart of (2) in this case is

$$v_{k+2} = L(v_k), \qquad v_0 \in V, \tag{4}$$

which generates the even terms of the original sequence $\{v_k\}$. Hence, by

letting $k' = k/2$, for k even, we have

$$v'_{k'+1} = L(v'_{k'}), \qquad v'_0 = v_0, \qquad k' = 0,1,\ldots, \tag{5}$$

where $v'_{k'} \equiv v_{2k}$. Since (5) is identical to (2), it readily follows that one sequence converges if, and only if, the other one does, and hence stability under the Gauss–Seidel procedure coincides with the notion of stability under the Jacobi iteration. Furthermore, any asynchronous computation where the players act at random points in time and use the most recently available policy information (regarding the other player) would again lead to an iteration of the type (5) (with a different, possibly random k', monotonically increasing in k a.s.) provided that no player remains idle for an infinite duration. Hence, the notion of asynchronously stable equilibrium coincides with that of stable equilibrium in two-person games, with iterations of the type above, regardless of the order of moves by the players.

When the policy spaces U, V are taken as finite-dimensional Euclidean spaces, a set of explicit conditions can be obtained for the convergence of (1), or equivalently (3), to the unique stable equilibrium solution, as given in Li and Başar (1987b). If the cost functions J^1 and J^2 are quadratic in their arguments, there will exist matrices C_1 and C_2 and vectors $d_1 \in U$, $d_2 \in V$, such that

$$L_1(v) = C_1 v + d_1, \qquad L_2(u) = C_2 u + d_2, \tag{6}$$

and the conditions alluded to above can be written as

$$\rho(C_1 C_2) < 1, \tag{7}$$

which is both *necessary and sufficient* for existence a stable equilibrium. The rate of convergence of algorithms (2) or (4) in this case would be $1/\rho(C_1C_2)$, and in general it will take an infinite number of steps to converge to the equilibrium solution.

The question we raise now is whether it is possible (i) to relax (7) and (ii) to improve the rate of convergence towards equilibrium by devising an algorithm which would still preserve some of the informational constraints imposed on the problem.

3. Relaxation algorithms for non-cooperative equilibria

The algorithms we propose here for on-line computation of non-cooperative equilibria are of the relaxation type, where each player is allowed to use some additional memory involving his most recent policy choice and/or past policies of the other player. More specifically, we replace the Jacobi-type

algorithm (3) with the 'relaxed' version

$$\begin{aligned} u_{k+1} &= \alpha u_k + (1-\alpha) L^1(v_k), \\ v_{k+1} &= \beta v_k + (1-\beta) L^2(u_k), \\ k &= 0,1,\ldots, \qquad u_0 \in U, \quad v_0 \in V, \end{aligned} \tag{8}$$

or the more general one

$$\begin{aligned} u_{k+1} &= \alpha u_k + (1-\alpha) L^1\big(\bar{\alpha} v_k + (1-\bar{\alpha}) v_{k-1}\big), \\ v_{k+1} &= \beta v_k + (1-\beta) L^2\big(\bar{\beta} u_k + (1-\bar{\beta}) u_{k-1}\big), \\ k &= 0,1,\ldots, \qquad u_0 \in U, \quad v_0 \in V, \end{aligned} \tag{9}$$

which also admits the version

$$\begin{aligned} u_{k+1} &= \alpha u_k + (1-\alpha)\bar{\alpha} L^1(v_k) + (1-\alpha)(1-\bar{\alpha}) L^1(v_{k-1}), \\ v_{k+1} &= \beta v_k + (1-\beta)\bar{\beta} L^2(u_k) + (1-\beta)(1-\bar{\beta}) L^2(u_{k-1}), \\ k &= 0,1,\ldots, \qquad u_0 \in U, \quad v_0 \in V, \end{aligned} \tag{10}$$

where in all cases $\alpha, \beta, \bar{\alpha}, \bar{\beta}$ are the scalars (design parameters) to be determined so that the given algorithms are convergent, with a fast rate of convergence.

In the domain of the Gauss–Seidel procedure, the counterpart of (8) would be

$$\begin{aligned} u_{k+1} &= \alpha u_k + (1-\alpha) L^1(v_k), \\ v_{k+1} &= \beta v_k + (1-\beta) L^2(u_{k+1}), \\ k &= 0,1,\ldots, \qquad u_0 \in U, \quad v_0 \in V, \end{aligned} \tag{11}$$

and that of (9) would be

$$\begin{aligned} u_{k+1} &= \alpha u_k + (1-\alpha) L^1\big(\bar{\alpha} v_k + (1-\bar{\alpha}) v_{k-1}\big), \\ v_{k+1} &= \beta v_k + (1-\beta) L^2\big(\bar{\beta} u_{k+1} + (1-\bar{\beta}) u_k\big), \\ k &= 0,1,\ldots, \qquad u_0 \in U, \quad v_0 \in V. \end{aligned} \tag{12}$$

We first note that in all cases above if the sequences generated by the respective algorithms converge, then the common limit point is the non-cooperative equilibrium solution. Second, we note that asynchronous implementation (due to delay in information exchange or failure in computation) will in general require different conditions for convergence [as opposed to what was observed in section 2 in the cases of (2) and (4)] since even if, for example, a

player does not receive any information from the other one for some lengthy period of time, he will still continue to iterate on his past choices, due to the presence of non-zero α or β.

To obtain some explicit results, we now restrict our analysis to games with quadratic cost functions, where the optimum reaction functions L_1 and L_2 are given by (6). With this, the algorithm (8) becomes the linear system

$$\begin{pmatrix} u_{k+1} \\ v_{k+1} \end{pmatrix} = \begin{pmatrix} \alpha I & (1-\alpha)C_1 \\ (1-\beta)C_2 & \beta I \end{pmatrix} \begin{pmatrix} u_k \\ v_k \end{pmatrix} + \begin{pmatrix} (1-\alpha)\, d_1 \\ (1-\beta)\, d_2 \end{pmatrix}. \tag{13}$$

Let the system matrix in (13) be denoted by $A(\alpha, \beta)$. Then, our earlier question, rephrased in this context, is the optimal choice of α and β so that $\rho(A(\alpha, \beta))$ is as small as possible, and at least $\rho(A(\alpha, \beta)) < 1$ whenever $\rho(A(0,0)) = \rho^{1/2}(C_1 C_2) \geq 1$.

Now, if the algorithm (13) is run asynchronously, a necessary and sufficient condition for its convergence is [Chazan and Miranker (1969), Baudet (1978)]

$$\rho(\bar{A}(\alpha, \beta)) < 1, \tag{14}$$

where

$$\bar{A}(\alpha, \beta) := \begin{pmatrix} \alpha I & |(1-\alpha)C_1| \\ |(1-\beta)C_2| & \beta I \end{pmatrix}. \tag{15}$$

Here, $|C|$ denotes the matrix derived from C by multiplying all its negative entries by -1, and we have also implicitly assumed that α and β are positive. Note that since A and $\bar{A}$ are in general different matrices, the presence of the diagonal terms will make the stability conditions in general different, thus corroborating our earlier remark (and observation) on this point.

Moving on to the generalized Gauss–Seidel procedure (11) for the case of quadratic functionals, it would read

$$\begin{aligned} u_{k+1} &= \alpha u_k + (1-\alpha)C_1 v_k + (1-\alpha)\, d_1, \\ v_{k+1} &= \beta v_k + (1-\beta)C_2 u_{k+1} + (1-\beta)\, d_2, \end{aligned} \tag{16}$$

and further substitution of the first relation into the second would lead to the first-order difference equation

$$\begin{aligned} \begin{pmatrix} u_{k+1} \\ v_{k+1} \end{pmatrix} &= \begin{pmatrix} \alpha I & (1-\alpha)C_1 \\ \alpha(1-\beta)C_2 & \beta I + (1-\alpha)(1-\beta)C_2 C_1 \end{pmatrix} \begin{pmatrix} u_k \\ v_k \end{pmatrix} \\ &\quad + \begin{pmatrix} (1-\alpha)\, d_1 \\ (1-\beta)[(1-\alpha)C_2\, d_1 + d_2] \end{pmatrix}. \end{aligned} \tag{17}$$

Hence, the problem here is to find the optimal values for α and β such that $\rho(B(\alpha,\beta))$ is minimized, where $B(\alpha,\beta)$ is the system matrix in (17). If, however, (16) were run asynchronously with $\alpha,\beta>0$, convergence would be guaranteed if, and only if [as the counterpart of (14) in this case], the following condition is satisfied:

$$\rho(\bar{B}(\alpha,\beta))<1, \tag{18}$$

where

$$\bar{B}(\alpha,\beta):=\begin{pmatrix} \alpha I & |(1-\alpha)C_1| \\ \alpha|(1-\beta)C_2| & |\beta I+(1-\alpha)(1-\beta)C_2C_1| \end{pmatrix}. \tag{19}$$

Here the implicit assumption has been that v_k in both (16) and (17) are restricted to be the same quantity, contributing to asynchrony in an identical manner.

In the fuller version of the paper, we work out in detail the conditions of convergence for various relaxation algorithms developed above, for some special cases. In each case it becomes apparent that the proposed algorithms offer considerable advantages (in terms of regions of convergence and rates of convergence) over the ones discussed in section 2.

References

Başar, T. and G.J. Olsder, 1982, Dynamic noncooperative game theory (Academic Press, London/New York).

Başar, T., 1986, A tutorial on dynamic and differential games, in: T. Başar, ed., Dynamic games and applications in economics, Lecture notes in economics and mathematical systems (Springer-Verlag, New York) 1–25.

Baudet, G.M., 1978, Asynchronous iterative methods for multiprocessors, Journal of ACM 25, 226–244.

Bertsekas, D.P., 1983, Distributed asynchronous computation of fixed points, Mathematical Programming 27, 107–120.

Chazan, D. and W. Miranker, 1969, Chaotic relaxation, Linear Algebra and Applications 2, 199–222.

Li, S. and T. Başar, 1987a, Asymptotic agreement and convergence of asynchronous stochastic algorithms, IEEE Transactions on Automatic Control AC-32, 612–618.

Li, S. and T. Başar, 1987b, Distributed algorithms for the computation of noncooperative equilibria, Automatica 23, 523–533.

Tsitsiklis, J.N., D.P. Bertsekas and M. Athans, 1986, Distributed asynchronous deterministic and stochastic gradient optimization algorithms, IEEE Transactions on Automatic Control AC-31, 803–812.