JOURNAL OF ECONOMIC THEORY 45, 342–352 (1988)

The Complexity of Computing Best-Response Automata in Repeated Games*

ITZHAK GILBOA

Department of Economics, Tel-Aviv University, Ramat Aviv, Tel-Aviv 69978, Israel

Received November 12, 1986; revised June 9, 1987

The following problem is examined: given a game and the opponents' finite automata, find a best-response automaton for a certain player in the repeated game. It is shown that the problem is relatively "easy" (i.e., of polynomial time complexity) if the number of players is fixed, but "difficult" otherwise. *Journal of Economic Literature* Classification Numbers: 021, 022, 026.

1. INTRODUCTION

Finite automata were introduced into the theory of repeated games in order to capture the somewhat vague concept of "bounded rationality" (originally introduced by Simon [7, 8] and formalized by Radner [5] as "ε-optimality"). In models such as Rubinstein [6], Neyman [4], and Kalai and Stanford [2] (the latter being restricted to finite cardinalities), finite automata are supposed to play the repeated games for the actual players, thus limiting the complexity of the available strategies. (The interpretation varies from one model to the other, but this seems to be the basic idea.)

In these models a Nash equilibrium is usually defined as a choice of an automaton for each player, such that no player can increase his payoff by a unilateral deviation. This definition implicitly assumes that the players are capable of computing a best-response automaton, given the other players' automata, or, at least, capable of judging whether a certain automaton (say, the one suggested to them by an $(n+1)$th party) is indeed a best-response automaton. However, this problem may be quite complicated. Even though it is not raised at every stage of the repeated game, but is rather solved "once and for all," its complexity is of interest to us since in reality the players usually play many repeated games simultaneously, and

* I thank Professor David Schmeidler for the discussions which motivated and directed this research. I am also grateful to two anonymous referees for helpful comments. Financial assistance from the Josef Buchman Fund and the use of typing facilities of the Foerder Institute for Economic Research are gratefully acknowledged.

0022-0531/88 $3.00

thus are frequently encountered with the best-response automaton problem.

This paper deals with the complexity of this problem. We shall see that it is relatively easy to solve when the number of players is fixed, but quite difficult otherwise. By "relatively easy to solve" we mean that there exists an algorithm which solves it in a polynomial time complexity, which is a common informal definition of a "simple problem."

If we believe that the number of players in a game is usually not known in advance, these results may be interpreted as a criticism of the Nash equilibrium concept in the context of repeated games with bounded rationality.

It should be stressed that we focus on the complexity of *computing* best-response automata and ignore the complexity of *implementing* them, while the models mentioned above have done the opposite. To some extent, then, our model is also too extreme and a synthesis of the two approaches seems to be called for. However, one should note that the computation and the implementation problems are not independent: If the computation of best-response automata is difficult, we have dubious justification to assume that best-response strategies are always used, even if they sometimes turn out to be easily implementable. On the other hand, if the computation is easy, its result has to be simple (for instance: the size of the automaton computed is bounded by the length of the computation)—hence the computed best-response strategy is also easy to implement.

We note that in the sequel the one-shot game is supposed to be given in its extensive form. Although mathematically cumbersome, the extensive form seems to be the appropriate one for analyzing complexity problems: First note that it is the more concise description of the game: a game in its normal form may be translated into an extensive form by an algorithm the time complexity of which is linear in the size of the original game. The converse is not true. In fact, the mere size of the game in the normal form can, in general, be bounded only by an exponential function of that of the original extensive-form game. Thus, if there exists a polynomial algorithm for the problem when phrased in the extensive-form terms, then there also exists one for the normal-form problem, but the converse is false.

Since we believe that games are originally given in their extensive form, and are translated into the normal form merely for mathematical convenience, we prefer to avoid this translation as soon as it ceases to be innocuous. As explained above, when complexity considerations are introduced, the translation from extensive to normal form does indeed change the nature of the game.

However, we should mention that not all the results presented in the sequel continue to hold if the extensive form is replaced by the normal one. In fact, Theorem A and C remain valid, while Theorem B does not. On the

other hand it should be emphasized that the qualitative results, as described above, are contained in Theorems A and C.

This paper is organized as follows: Section 2 presents the model. It includes formal definitions of all computer science terms used in this paper, the one exception being "NPC problems." On account of its length, the formal definition of this term is substituted for an informal explanation plus a reference. Section 3 contains the formal definitions of the problems discussed and the statement of the results (concerning the complexity of these problems.) Section 4 contains the proofs of the theorems.

One more technical remark is required here: As seems to be prevalent in game theory literature, this paper refers to the players as if they were known to be men (rather than women). No significance should be attached to this convention.

2. Definitions and Notations

A *game* (in its extensive form) is a septuple $G = (n, Z, \{u_i\}_{i=1}^n, \{Y_i\}_{i=1}^n, \{\Pi_i\}_{i=1}^n, \{B_i\}_{i=1}^n, \mu)$, where

(1) n is the number of players, henceforth denoted by $N = \{1, ..., n\}$.

(2) Z is a finite and non-empty set of *outcomes* of the game.

(3) $u_i: Z \to R$ is player i's utility function.

(4) Y_i is a finite and non-empty set of the *decision nodes* of player i.

(5) Π_i is a partition of Y_i into $Y_i^1, ..., Y_i^{k_i}$, each of which is non-empty. (Π_i is the *information partition.*)

(6) B_i is a finite and non-empty set (of *acts* of player i) which is partitioned into $B_i^1, ..., B_i^{k_i}$, all of which are non-empty. (B_i^j is the set of possible acts of player i at each of the nodes in Y_i^j.)

(7) $\mu: \{(Y_i^j, b) \mid b \in B_i^j,\ i \leqslant n,\ j \leqslant k_i\} \to Y \cup Z \equiv X$ is such that the graph $\mathrm{Gr}(X, E)$, where

$$E = \{(y, \mu(Y_i^j, b)) \mid y \in Y_i^j, b \in B_i^j,\ i \leqslant n,\ j \leqslant k_i\}$$

forms a tree (the leaves of which are Z).

A *strategy* for player i in a game G is a function $s_i: \{Y_i^j\}_{j \leqslant k_i} \to B_i$ such that $s_i(Y_i^j) \in B_i^j$. The set of all strategies of player i will be denoted S_i.

A *repeated game* is an infinite repetition of the one-shot game. If $z_t \in Z$ is the outcome of the game at stage t, the utility of player i in the repeated game is

$$U_i(z_1, z_2, ...,) = \liminf_{t \to \infty} \frac{1}{t} \sum_{\tau=1}^{t} u_i(z_\tau).$$

We note that the specification of U_i is of little importance. In fact, all the results presented in this paper are also valid for any other reasonable utility function. (That is to say, a function U_i for which our results do not hold is unreasonable by definition. The limsup of the average payoff and the discounted utility are, however, "reasonable" according to this definition.)

A finite automaton for player i in a game G is a quadruple $A = (Q, q_0, \delta, \lambda)$ such that Q is a finite and non-empty set of *states*, $q_0 \in Q$ is the *initial state*, $\delta: Q \times Z \to Q$ is the *transition function*, and $\lambda: Q \to S_i$ is the *behaviour function*.

Given a game G and a finite automaton $A_i = (Q^i, q_0^i, \delta^i, \lambda^i)$ for each player i in G, the repeated game is played as follows: at stage 1 each player i plays $\lambda^i(q_0^i)$. After stage $t \geqslant 1$ was played, yielding the outcome z_t, each automaton A_i moves from the state q_t^i to $\delta^i(q_t^i, z_t) \equiv q_{t+1}^i$ and player i plays $\lambda^i(q_{t+1}^i)$ at stage $t+1$.

A finite automaton A_i is a *best-response* automaton for player i in a game G versus $\{A_j\}_{j \neq i}$ if it maximizes U_i (player i's utility in the repeated game) subject to the constraint that any other player $j \neq i$ plays according to A_j.

We now turn to discuss the concept of a "problem." In computer science, this term usually refers to a YES/NO problem (such as: "Does there exist a Hamiltonian path in a given graph?"). These problems are formally defined as subsets of the set of all possible input data: a "problem" is identified with the set of inuts for which the answer is "YES." However, we shall use this term in a wider sense, which will include such problems as "Find the maximal path in a given graph" and so forth. We will not formally define a "problem" and such a definition is unnecessary: throughout this paper we refer only to (a few) specific problems, so that they may be considered merely as textual conventions, rather than formal mathematical objects. For instance, if we define the problem P to be: "Find the sum of two given numbers," we may state a theorem which says that there exists an algorithm which solves P. This should be read as: "There exists an algorithm which finds the sum of two given numbers."

By the term "*a complexity* of an algorithm" we shall refer to the time complexity, that is, the maximal number $c(n)$ of operations that have to be performed for n given input items. As is usually done, we will not focus on the function $c(n)$ itself but rather on its order of magnitude $O(c(n))$. More specifically, we will be interested in the existence of "polynomial algorithms," i.e., algorithms for which the time complexity $c(n)$ is bounded from above by some polynomial of n.

We shall use the concept "NP-complete" (or "NPC") as defined in Aho *et al.* [1]. (A formal definition of the term is too long and cumbersome to be cited here.) A reader who is not acquainted with the term should only know the following facts: A problem is NP if, roughly speaking, it can be solved in a polynomial time complexity by an unlimited number of

processors. A problem P is NPC if the following conditional statement is true: *if* there were a polynomial algorithm which solves P, *then* there would be such an algorithm for any NP problem. The class of NP problems is of great importance since it includes many combinatorical problems arising in various applications. Most of these problems turn out to be NPC. Unfortunately, none of the NPC problems is known to be solvable by a polynomial algorithm. On the other hand, it has not yet been proved that an NPC problem cannot be solved by such an algorithm. However, for all practical purposes we may consider NPC problems as problems for which there is no polynomial algorithm.

3. Statement of Results

We are interested in the problem of computing a best-response automaton versus given automata of the other players. A related YES/NO problem is: Is a given automaton for player i a best-response automaton versus other given automata? Or: Is an n-tuple of automata a Nash equilibrium?

Both problems may be posed for a predetermined number of players. We thus end up with four problems:

(1) BRA_n (*best-response automaton*): Given a game G with n players, and n finite automata for the players in G, is A_1 a best-response automaton for player 1 versus $(A_2, ..., A_n)$?

(2) BRA'_n: Given a game G with n players and $(n-1)$ automata $(A_2, ..., A_n)$ for players 2, ..., n, find a best-response automaton for player 1.

(3) BRA: as BRA_n, with no restriction on the number of players in G.

(4) BRA': as BRA'_n, with no restriction on the number of players in G.

The results concerning these problems are:

THEOREM A. *For each* $n \geq 1$ *there exist polynomial algorithms which solve* BRA_n *and* BRA'_n.

THEOREM B. *If there exists a polynomial algorithm which solves* BRA *there exists a polynomial algorithm for any* NPC *problem.*

THEOREM C. *There does not exist a polynomial algorithm which solves* BRA'.

4. Proof of the Theorems

4.1. *Proof of Theorem A*

We begin by constructing an algorithm which solves BRA'_n for a given n. Let there be given a game G and $(n-1)$ automata $\{A_j\}_{j>1}$ for players 2, ..., n. Denote the total number of input items by M. M will be used as an upper bound for the size of the game tree ($|X|$), the size of the automata's states sets ($|Q_i|$), and so forth.

Construct a directed graph $\mathrm{Gr}_0(V_0, E_0)$, where $V_0 = Z \times Q_2 \times \cdots \times Q_n$ and E_0 is defined as follows: there is an edge from $(z, q^2, q^3, ..., q^n)$ to $(\bar{z}, \bar{q}^2, \bar{q}^3, ..., \bar{q}^n)$ iff (i) there exists a strategy s^1 of player 1 such that, if he plays s^1 and every other player $i \geqslant 2$ plays $\lambda^i(q^i)$, the outcome of the game is $\bar{z}$; (ii) for all $i \geqslant 2$, $\delta^i(q^i, \bar{z}) = \bar{q}^i$.

Let $\overline{\mathrm{Gr}_0}$ denote the subgraph of Gr_0 which contains only the nodes that can be reached from one (therefore all) of $\{(z, q_0^2, q_0^3, ..., q_0^n) \mid z \in Z\}$.

Note that Gr_0 may be constructed in polynomial time: $|V_0|$ is bounded by M^n, and the construction of E_0 may be done in $O(M^{2n+1})$ operations: for each pair of vertices v_1, v_2 in V_0, only $O(M)$ operations are required to determine whether $(v_1, v_2) \in E_0$. (As the strategies of players 2, ..., n are known to be $\lambda^i(q^i)$, one does not have to scan all possible strategies of player 1 in order to check whether condition (i) is satisfied. It suffices to scan the path from the root of the game tree to the node $\bar{z}$, and this path length is bounded by M.)

Now let A_1 be any finite automaton for player 1 in G. Consider the graph $\mathrm{Gr}_1(V_1, E_1)$, where $V_1 = Z \times Q_1 \times Q_2 \times \cdots \times Q_n$ and E_1 is defined as follows: there is an edge from $(z, q^1, q^2, ..., q^n)$ to $(\bar{z}, \bar{q}^1, \bar{q}^2, ..., \bar{q}^n)$ iff (i) if each player $i \in N$ plays $\lambda^i(q^i)$, then the outcome of the game is $\bar{z}$; (ii) for each $i \in N$, $\delta^i(q^i, \bar{z}) = \bar{q}^i$.

It is both trivial and well known that when $A_1, ..., A_n$ play the repeated game, they move in a simple cycle C_1 in Gr_1 (i.e., a cycle which has no subcycles). Consider the cycle C_0 in Gr_0 induced by C_1. C_0 need not be a simple cycle, but it may be decomposed into simple cycles $C_0^1, ..., C_0^r$. Without loss of generality, assume that C_0^1 is the one which maximizes player 1's average payoff. (This criterion must, of course, vary with the definition of U_i). Hence, if A'_1 is an automaton for player 1 such that only C_0^1 is repeated when $A'_1, A_2, ..., A_n$ play the repeated game, then A'_1 is at least as good for player 1 as A_1.

It is, however, very simple to construct such an automaton A'_1. It will have $|C_0^1|$ states, each of which corresponding to a vertex in C_0^1 (and visited infinitely often in the play of the game), and up to $|V_0|$ states corresponding to the minimal path from the initial state to C_0^1. (Such a path exists since there was a corresponding one in C_1.)

It therefore suffices to find the best automaton for player 1 in the restricted class of automata which correspond to cycles in $\overline{\mathrm{Gr}_0}$, the length of which does not exceed $|V_0|$. (This class includes the simple cycles.) Since the construction of the automaton for a given cycle of this class is easily performed in polynomial time, the problem BRA'_n may be reduced to the problem of finding the average-utility maximal cycle in a graph, the length of which does not exceed $|V_0|$. However, this problem is of polynomial complexity, as proved in Lawler [3, pp. 94–97]. This completes the proof of the first part of the theorem.

We now turn to the second part, namely, that there exists an algorithm which solves BRA_n in polynomial time complexity. Given the game G and n automata $A_1, ..., A_n$, we follow the play of the game. (This is tantamount to drawing a path in Gr_1.) After up to $|V_1|$ stages we recognize the cycle that will be repeated eternally, and we may easily compute player 1's utility in the repeated game. Then we use the algorithm which solves BRA'_n, find the best-response automaton A'_1 for player 1 versus $A_2, ..., A_n$, and compare the payoffs guaranteed by A_1 and A'_1. A_1 is a best-response automaton iff its payoff equals that of A'_1. This completes the second half of the proof. Q.E.D.

Remarks. (1) Note that the maximal cycles computed in the proof are *not* necessarily *simple* cycles. The problem of finding a maximal *simple* cycle is, of course, at least as hard as (hence equivalent to) the problem of existence of an Hamiltonian cycle, and to the best of our knowledge, it cannot be solved by a similar polynomial algorithm.

(2) Note that if the number of players is not known in advance (as is the case in BRA and BRA'), the algorithms presented above are of exponential complexity. In the next section we shall see that these algorithms cannot be improved significantly.

4.2. *Proof of Theorem B*

We shall show that the problem of existence of a Hamiltonian cycle is reducible to BRA. That is, that if there were a polynomial algorithm solving BRA, there would also be one solving the Hamiltonian cycle existence problem.

Assume, then, that there exists a polynomial algorithm solving BRA. Let an undirected graph Gr be given. We shall show that one may construct in polynomial time, a game G and a finite automaton A_i for each player in G, such that A_1 is a best-response automaton for player 1 in G iff there is no Hamiltonian cycle in Gr.

Assume $\mathrm{Gr} = (V, E)$, where $V = \{v_1, ..., v_n\}$. We first define a new graph $\mathrm{Gr}' = (V', E')$, where $V' = V \cup \{w_1, ..., w_n\}$ and

$$E' = E \cup \{(w_i, w_{i+1}) \mid 1 \leqslant i \leqslant n-1\}.$$

That is, Gr′ is the disjoint union of two graphs: the original Gr and an additional one, in which there is an $(n-1)$-long path from w_1 to w_n but no other edges.

We now define the one-shot game. There are $n+3$ players. The game is played in $(n+3)$ turns with complete information as follows:

(1) In the first $(n+1)$ turns players 3, ..., $n+3$ play. Each player $i \geqslant 3$ may either use his veto strategy, in which case the game ends, or refrain from doing so and let the game continue. If player i has chosen his veto strategy, the outcome of the game is $p_i \in Z$.

(2) In turn $(n+2)$ player 2 is asked to choose a vertex from Gr′. Suppose he chose $y_2 \in V'$.

(3) In turn $(n+3)$ player 1 is asked to choose a vertex from Gr′. Suppose he chose $y_1 \in V'$. The outcome of the game is now determined to be $(y_2, y_1) \in Z$.

To complete the description of the game, we have to specify the utility functions. However, only u_1 is relevant to BRA, so that u_i may be identically zero for $i > 1$. Player 1's utility is given by

$$u_1(p_i) = -1 \qquad u_1((y_2, y_1)) = 1 \qquad (y_2, y_1) \in E'$$
$$= 0 \qquad \text{otherwise.}$$

That is to say: player 1 is punished by any veto strategy of players 3, ..., $n+3$. If none of these players has vetoed, player 1's utility is determined by his ability to find a vertex which is connected to the one player 2 has chosen.

Next we describe $(n+3)$ finite automata, one for each player. Begin with player 3. His automaton has exactly four states, and it plays a "trigger strategy": it chooses the benevolent strategy (not to veto) as long as none has vetoed and the outcomes of the game in the two last stages (y_2^{t-1}, y_1^{t-1}) and (y_2^{t-2}, y_1^{t-2}) are such that $y_1^{t-1}, y_1^{t-2} \in V$ or $y_1^{t-1}, y_1^{t-2} \in V' - V$. If this is not the case then player 3's automaton chooses to veto. In fact, player 3 guarantees that player 1 will always choose one of the vertices $\{v_1, ..., v_n\}$ or that he will stick to $\{w_1, ..., w_n\}$, but will not alternate between the two sets of vertices.

Now consider player $3+i$ for $1 \leqslant i \leqslant n$. This player's automaton has $(n+1)$ states and it plays as follows: it "counts" the number of stages that have passed since player 1 has chosen one of $\{v_i, w_i\}$ for the last time. If player 1 chooses one of these strategies again after less than n stages, player i's automaton decides to punish player i and veto.

Player 2's automaton is a "mirror" automaton: if the previous stage outcome is (y_2^{t-1}, y_1^{t-1}), it will play $y_2^t = y_1^{t-1}$. Otherwise (if the previous stage

outcome is p_i for some i, or that $t=0$ and there is no previous stage) it chooses an arbitrary vertex, say v_1. Such an automaton requires only $2n$ states.

We are left with player 1's automaton A_1. Let this be an n-state automaton, which plays w_i at any stage $t \equiv i \pmod{n}$.

We now claim that A_1 is a best-response automaton versus $A_2, ..., A_{n+3}$ iff there is no Hamiltonian cycle in Gr. To verify this, we first observe that a best-response automaton for player 1 has to be one which chooses *either* a permutation of $\{v_1, ..., v_n\}$ *or* a permutation of $\{w_1, ..., w_n\}$ and chooses the vertices according to the cyclic order induced by this permutation. It is obvious that any other strategy will result in eternal punishment: player 3 does not allow to mix v's and w's. The choice of v's (or w's) in the first n stages is left to player 1, as long as he does not repeat a choice within these n stages. Afterwards he has to repeat the whole cycle in the same order, or else he will be punished by some player $3+i$ $(1 \leqslant i \leqslant n)$. Note that A_1 assures player 1 a payoff $U_1 = 1 - 1/n$. It is a best-response automaton iff the payoff 1 cannot be achieved. As the maximal path in $V' - V$ is of length $(n-1)$, this is equivalent to the non-existence of a Hamiltonian cycle in the original graph Gr.

To complete the proof we note that the construction of the game G and the automata for the players can be carried out in polynomial time complexity: G has n^2+n+1 outcomes, and the $(n+3)$ automata all have $O(n)$ states, hence the description of the transition function δ of each of them is $O(n^3)$. Q.E.D.

Remark. In a similar way it may be proved that the following problem: "Given a game G, $(n-1)$ automata for players 2, ..., n, and a number k—is there an automaton for player 1 which assures the payoff k?"—is also at least as hard as NPC problems. (In fact, one may use the very same proof with player 1's payoffs multiplied by k.)

4.3. *Proof of Theorem C*

The proof is based on an example. For each $n \geqslant 0$ we construct a game G_n with $(n+1)$ players, and n automata for players 2, ..., $n+1$, such that the best-response automaton for player 1 has to have a number of states which cannot be bounded by any polynomial of the size of the input. In particular, the description of such an automaton cannot be written down in polynomial time complexity.

We begin with the game G_n: it is played in $(n+1)$ turns as follows: in the turn $1 \leqslant i \leqslant n$ player $(i+1)$ chooses one of the numbers $\{0, 1\}$. In the last turn, player 1 has to guess all the other player's choices (of course, without knowing what these choices were). If player 1 guesses correctly all

n numbers, he wins the payoff 1. Otherwise, his payoff is 0. (Again, the payoffs of the other players are irrelevant.)

Note that the game G_n has $k \equiv 2^{2n}$ outcomes. Next we describe the automata: for each $i \geq 2$, player i uses an automaton A_i which, regardless of its input, plays 1 every $(k-i+2)$ stages, and plays 0 at all other stages. We now have the following:

Claim. If, in the game G_n, player $i \geq 2$ plays 1 exactly at those stages t such that $t \equiv 0 \pmod{m_i}$, then a best-response automaton for player 1 has at least lcm $(m_2, ..., m_{n+1})$ states (lcm = least common multiplication).

Proof. For each t, let v_t be the vector of residuals: $v_t = (t \bmod m_2, ..., t \bmod m_{n+1})$. Of course, there are exactly lcm $(m_2, ..., m_{n+1})$ different vectors in $\{v_t\}_{t \geq 0}$. Suppose now that, contrary to our claim, the best-response automaton A_1 has less than lcm $(m_2, ..., m_{n+1})$ states. Hence, there are two different residual vectors $v^1 \neq v^2$ and a state $q \in Q^1$ such that during the play of the game it happens that A_1 is at q while the other players' automata states are given by v^i (for $i = 1, 2$). Consider the entries in v^1 and v^2 which are zero. If there exists an $i \geq 2$ such that $v^1(i) = 0$ and $v^2(i) \neq 0$ (or vice versa), then player i's behaviour is not the same in both cases. (In one case he chooses 1 while in the other he chooses 0.) No matter what $\lambda^1(q)$ is, it cannot simultaneously guess correctly what player i is going to do, and consequently A^1 is *not* a best-response automaton. Assume, then, that $v^1(i) = 0$ iff $v^2(i) = 0$. This implies that all the players choose exactly the same strategies in both cases. We may therefore consider the next stage, in which, again, there will be two different residual vectors $\bar{v}^1$ and $\bar{v}^2$ and a state $\bar{q} \in Q^1$ such that during the play of the game it happens that A is at $\bar{q}$ while the others' automata states are described by $\bar{v}^i$ (for $i = 1, 2$). Continuing inductively up to $\min_{2 \leq i \leq n+1} m_i$ steps, a contradiction is proved.

To complete the proof of the theorem, all we have to show is that lcm $(k-n+1, ..., k)$ cannot be bounded from above by any polynomial of k. (The total number of input items is $O(k^2 lgk)$: the game's description is $O(k)$, and there are $n = (1/2)\, lgk$ finite automata, each having $O(k)$ states and k possible inputs. Thus the total number of input items is bounded by $k^3 \equiv M$. If there were a polynomial algorithm in M, it would also be polynomial in k.)

Let there be given an integer $p \geq 0$. There exists a function f such that

$$\text{lcm } (k-n+1, ..., k) \geq \frac{(k-p+1) \cdot ... \cdot k}{f(p)} \qquad \text{if} \quad n \geq p.$$

(In fact, $f(p) \leq 2^{p/2} \cdot 3^{p/3} \cdot ... \cdot p$. This bound is obtained as follows: the lcm of n given numbers may be obtained by multiplying all of them, and

then dividing the result by common divisors. In the sequence of the p consecutive numbers preceding k, at most $p/2$ may have 2 as a divisor, at most $p/3$ may be divided by 3, and so forth.)

For a constant p, there exists a polynomial of k of degree p which is a lower bound of lcm $(k-n+1, ..., k)$. This being true for all $p \geqslant 0$, there is no polynomial in k which can bound lcm $(k-n+1, ..., k)$ from above.

Q.E.D.

REFERENCES

1. A. V. AHO, J. E. HOPCROFT, AND J. D. ULLMAN, "The Design and Analysis of Computer Algorithms," Addison-Wesley, Reading, MA, 1974.
2. E. KALAI AND W. G. STANFORD, "Equally Sophisticated Players: On the Complexity, Memory and Automation of Repeated Game Strategies," mieo, 1985.
3. E. LAWLER, "Combinatorial Optimization: Networks and Matroids," Holt, Rinehart & Winston, New York, 1976.
4. A. NEYMAN, "Bounded Complexity Justifies Cooperation in the Finitely Repeated Prisoner's Dilemma," mimeo, 1985.
5. R. RADNER, "Can Bounded Rationality Resolve the Prisoner's Dilemma?" mimeo, 1978.
6. A. RUBINSTEIN, Finite automata play—The repeated prisoner's dilemma, *J. Econ. Theory* **39** (1986), 83–96.
7. H. A. SIMON, Theories of Bounded Rationality, *in* "Decision and Organization" (C. B. McGuire and R. Radner, Eds.), North-Holland, Amsterdam, 1972.
8. H. A. SIMON, On how to decide what to do, *Bell J. Econ.* **9** (1978), 494–507.