A NOTE ON STRATEGY ELIMINATION IN BIMATRIX GAMES

Donald E. Knuth[1], Christos H. Papadimitriou[1], and John N. Tsitsiklis[2]

ABSTRACT: *In bimatrix games we study the process of successively eliminating strategies which are dominated by others. We show how to perform this simplification in $O(n^3)$ time, where n is the number of strategies. We also prove that the problem is P-complete, which suggests that it is inherently sequential.*

1. INTRODUCTION

A *two person* (or *bimatrix*) game is a pair of $m \times n$ matrices A, B, with integer entries. This game is played between two players A and B. Player A chooses a row i, player B simultaneously chooses a column j. As a result, A receives a_{ij} (dollars, say), and B receives b_{ij}.

An easy way to simplify a bimatrix game is to eliminate from both A and B any strategy (row or column) that is *dominated* by another. We say that strategy i of player A dominates strategy i' of the same player if $a_{ij} \geq a_{i'j}$ for $j = 1, \ldots, n$. Similarly, strategy j of player B dominates strategy j' of the same player if $b_{ij} \geq b_{ij'}$ for $i = 1, \ldots, m$. Obviously, both player A and B can avoid playing dominated strategies, with no deterioration of the outcome. Thus, all dominated strategies can be eliminated. Furthermore, deletion of dominated strategies may result in new domination, and so on, until we are left with a *reduced* game, in which no further elimination is possible. Strategy domination, and its obvious generalization to many players, has long been a well-known notion of simplification in games. More recently, a new concept of rationality was proposed, based on strategy domination [Be, Pe]. In this note we look at the algorithmic aspects of the process of producing a reduced game.

The obvious algorithm for doing this is to repeatedly test each pair of strategies for domination, and to delete the strategies that were found to be dominated. Since there are $O(m^2 + n^2)$ pairs to be examined, each test entails checking n (or m) inequalities, and the whole process must be repeated after each row or column deletion, the most straightforward algorithm takes $O(mn(m+n)^2) = O(m+n)^4$ time overall. But there is a faster ($O(m+n)^3$) algorithm, based on a rather well-known technique that can be called "obstruction counts". For each pair of strategies, we can maintain and update a count of the opponent strategies that prevent the first from dominating the second.

We do not know how to improve this algorithm asymptotically. In fact, it is not at all obvious how to determine in $o((m+n)^3)$ time whether there is *any* domination (in other words, whether or not the given game is *already* reduced). We show that this can be done asymptotically faster in the case of $0-1$ entries of the matrices, by reducing the problem to matrix multiplication. This technique, however, does not appear to extend to the case of integer entries, or to the problem of finding the final reduction.

We also discuss the issue of whether the process of reducing a bimatrix game can be satisfactorily solved *in parallel*. In general, a parallel algorithm is considered to be

[1]Department of Computer Science, Stanford University.

[2]Department of Electrical Engineering and Computer Science, M.I.T.

satisfactory if the time delay is polynomial in the *logarithm* of the length of the input, and the number of processors used is polynomial in the length of the input. The class of problems solvable with such algorithms is called NC. It is clear that NC is a subset of P. The important question for parallel computation, analogous to the P=?NP enigma for sequential computation, is whether NC=P. That is, while P=?NP asks whether there are problems in NP that are inherently exponential, NC=?P asks whether there are problems in P that are *inherently sequential.* It is considered likely that such problems do exist, but no proof has been found.

All algorithms that we know for finding the reduced game of a given bimatrix game are incapable of dramatic parallel speedups, since they wait for a strategy to be eliminated before eliminating the ones triggered by its deletion. Is this an inherent difficulty of the problem? Certain other problems for which only iterative algorithms are known (linear programming, maximum flow, or Markov decision-making), have been shown *P-complete:* A problem is P-complete if all other polynomially solvable problems are reducible to it in logarithmic space. Among all problems in P, the P-complete problems are the least likely to be in NC, and thus they are most probably inherently sequential. In the last section we show that the problem of computing the reduced game of a given bimatrix games is indeed P-complete. By contrast, testing whether a game is already reduced is trivially in NC. Our positive results hold for arbitrary bimatrix games, whereas our negative result is true even for *zero-sum* games (games such that $A + B = 0$).

2. THE CUBIC ALGORITHM

Before we describe our $O(m+n)^3$ algorithm, let us note that there may be several choices for strategies (rows and columns) to eliminate, and different sequences of deletions result in different reduced games. For example, even in the zero-sum game $\begin{pmatrix} 1 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix}$ we can get either $\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$ or $\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$ as the reduced game, depending on the precise sequenceof eliminations. However, the following reassuring result shows that these differences are superficial:

Proposition 1. In a bimatrix game, the final reduced game is unique up to row and column permutation.

Proof: Suppose that at some point during the deletion process a row i can be deleted (presumably because it is dominated by row j). Naturally, we may choose not to delete it immediately. We claim that either i is eventually deleted, or another row i' is deleted which, when restricted to the columns that have not been deleted at the time of i''s deletion, is identical to row i. For if row i is not eventually deleted, column j must be deleted. (Column deletion does not harm row dominance.) However, the row j^1 that caused $j = j^0$ to be deleted also dominates i, so it must also be deleted. Similarly, the row j^2 that dominates j^1 also dominates i, and so on. The only possibility that may eventually save i is for i itself to delete some row j^ℓ in this sequence. But this means that j^ℓ was at the time of its deletion equal to i (since it both dominated i and was dominated by it). The same argument holds for the columns. □

Our algorithm for finding the reduced matrix is based on the following idea: For each pair of rows (i,j), let r_{ij} be the number of columns k such that $a_{ik} > a_{jk}$; in other words, r_{ij} is the number of columns which prevent row j from dominating row i. Similarly, c_{ij} is the number of rows that prevent column j from dominating column i in matrix B. Notice that these matrices can be computed in $O(mn(m+n))$ time in the obvious way.

Once the R and C matrices are initialized, the algorithm works in stages. At each stage we examine both R and C matrices to discover a zero entry (this can be done in $O(m^2+n^2)$ time). If no such entry exists, the matrix is reduced and the algorithm terminates. If on the other hand such an entry is found (say, $r_{ij}=0$), this means that row j currently dominates i, and thus row i can be deleted. Once row i is deleted, we must update the matrix C, since perhaps row i was one reason for which a certain column failed to dominate another. In order to update $c_{k\ell}$ (for columns k and ℓ which have not been already deleted) we simply subtract one from $c_{k\ell}$ iff row i was indeed one of the reasons for which ℓ failed to dominate k, that is, iff $b_{i,k} > b_{i,\ell}$. Notice that the updating takes time $O(n^2)$, or $O(m^2)$ if a column was deleted. Hence the total stage can be performed in $O(m+n)^2$. Since there are at most $m+n$ stages, we have:

Theorem 2. The reduction of a bimatrix game can be computed in time $O(m+n)^3$. □

The problem of telling whether or not a given game is already reduced appears to be significantly simpler. In this problem there is no interaction between rows and columns, and thus we need only examine the problem of determining whether any row in a given $n \times m$ matrix A dominates another. In fact, consider first the special case in which the entries of A are 0 or 1. In this case the value of r_{ij} defined by the cubic algorithm is the number of columns k such that $a_{ik}=1$ and $a_{jk}=0$, and we may write

$$\overline{r_{ij}} = \sum_{k=1}^{n} a_{ik}\overline{a_{jk}}$$

where $\overline{a} = 1-a$. This is equivalent to the matrix equation

$$R = A\overline{A}^T;$$

hence the problem can be solved by matrix multiplication in time $O(m+n)^\alpha$, where α is the matrix multiplication exponent (currently known to be at most $2.375\ldots$ [CW]).

This can be generalized to the case in which the entries of matrix A belong to a set of r distinct values, say $\{0,1,\ldots,r-1\}$. We define the matrices C^k and D^k, $k=0,\ldots,r-1$, as follows:

$$c_{ij}^k = \begin{cases} 1, & \text{if } a_{ij}=k; \\ 0 & \text{otherwise,} \end{cases}$$

and

$$d_{ij}^k = \begin{cases} 1, & \text{if } a_{ij}<k; \\ 0 & \text{otherwise.} \end{cases}$$

Then clearly

$$R = \sum_{k=1}^{r-1} C^k D^k,$$

and the computation requires $r-1$ matrix multiplications. This is not very exciting, however, as r can be as large as mn.

3. P-COMPLETENESS

We shall show that the problem of computing the reduction of a zero-sum game is P-complete by reducing the *circuit value problem* (CVP) to it. A *circuit* is a finite sequence of triples $C = ((x_i, y_i, z_i), i = 1, \ldots, k)$. For each $i \leq k$, x_i is one of the "operations" **false**, **true**, **and**, **or**, and **not**; the other components y_i and z_i are nonnegative integers smaller than i. If a_i is either **false** or **true**, the triple is called an *input*, and $y_i = z_i = 0$. If x_i is either **and** or **or** then the triple is called a *gate*, and $y_i, z_i > 0$. The *value* of a triple is defined recursively as follows: The value of an input (**true**, 0, 0) is **true**, and the value of (**false**, 0, 0) is false. The value of a gate (x_i, y_i, z_i) is the Boolean operation denoted by x_i applied to the values of the y_ith and z_ith triples. The value of the circuit C is the value of the last gate. Finally, the CVP is the following problem: Given a circuit C, is its value **true**? It is known [Co] that this problem is P-complete even if (a) there are no **not** gates; (b) all **and** gates have **or** gates as inputs; and (c) **or** gates have **and**, **false**, or **true** gates as inputs.

Theorem 3. Given a zero-sum game, the problem of deciding whether a particular strategy is eliminated in some reduced game is P-complete.

Proof: We know already that the problem is in P. For P-completeness, we shall reduce the CVP to it. Given a circuit C we shall construct a zero-sum game A ($B = -A$) such that row 1 of A is deleted in the reduced game if and only if the value of C is **true**. Suppose that C has n **or** gates, and m triples of the remaining kinds. Then A is an $(m+1) \times (n+m)$ matrix. The first m rows correspond to the m **and**, **true**, and **false** gates (assume that the output gate is an **and** gate, corresponding to the first row of the matrix), and the first n columns to the n **or** gates. The entries will be such that a line (row or column) can be deleted if and only if the corresponding triple has value **true**.

We shall first describe the submatrix of the entries $a_{ij}, 1 \leq i \leq m$ and $1 \leq j \leq n$. We have $a_{ij} = 0$ in this region, except of the following cases: (a) One of the two operands of the **and** gate that corresponds to row i is the **or** gate corresponding to column j; in this case $a_{ij} = 3$. (b) One of the two operands of the **or** gate that corresponds to column j is the gate corresponding to row i; in this case $a_{ij} = -3$. (c) If row i corresponds to a **false** gate, then $a_{ij} = 2$. Thus, a line corresponding to an **and** and **or** gate is all 0's, except that it has 3's and -3's at the positions corresponding to the gates adjacent to it. Rows corresponding to **true** inputs are all 0's, and rows corresponding to **false** inputs are all 2's in the first n column positions, excepti for -3's where they are inputs.

The last of row matrix A is all 1's: $a_{m+1,j} = 1$ for $j = 1, \ldots, m+n$. This means that row $m+1$ dominates any row corresponding to an **and** gate *provided that its two columns corresponding to its two* **or** *inputs have been deleted*, and with them the obstructing 3's. Also, row $m+1$ dominates the **true** inputs, and fails to dominate the **false** inputs. Notice that this is precisely what we need in order to support the correspondence between a deleted row and a **true** gate.

The entries $a_{i,n+j}, i,j = 1,\ldots m$ are such that one of the last m columns dominates one of the first n columns if and only if *the row corresponding to at least one of the inputs of the corresponding* or *gate has been deleted.* This is done simply by letting

$$a_{i,n+j} = \begin{cases} 1, & \text{if } i = j; \\ -1 & \text{otherwise.} \end{cases}$$

(Recall that, since $B = -A$, column j dominates column j' if $a_{ij} \leq a_{ij'}$ for all i.) This completes the construction of the matrix A.

We claim that a line of A corresponding to a triple $g_\ell = (x_\ell, y_\ell, z_\ell)$ is deleted if and only if g_ℓ has value **true**[1]. The proof is by induction on ℓ. For the basis, if the deleted line is a row corresponding to a constant, then it cannot be a **false**, since such rows can never be eliminated, and a row corresponding to a **true** input is always dominated by the last row. If g_ℓ is an **and** gate, then clearly its row is deleted if and only if it was dominated by the last row (assuming that there are no two **and** gates with the same input in C) which happens if and only if both columns at which this row has a 3 have been deleted. By induction, this is equivalent to saying that both of the operants of g_ℓ have value **true**, and thus so does the gate in question. Finally, if g_ℓ is an **or** gate, the corresponding column is deleted if and only if it is dominated by one of the m last columns. This, however, means that there are no two rows at which this column has a -3, which happens if and only if at least one of these two rows has been eliminated. By induction, the corresponding gate had value **true**, and thus so does the **or** gate.

It follows that row 1 of the matrix will be eliminated if and only if the value of circuit C was **true**. Finally, it is clear that the above reduction can be carried out in logarithmic space (not counting input and output), since it only involves calculations on the indices (i, j, ℓ, etc.) of the gates and the lines of the matrix; such calculations can be done in logarithmic space. □

REFERENCES

[Be] D. Bernheim "Rationalizable Strategic Behavior", *Econometrica*, 52, pp. 1007-1028, 1985.

[Co] S.A. Cook "Towards a Complexity Theory of Synchronous Parallel Computation" *Enseign. Math. 2*, 27, pp. 99-124, 1981.

[CW] D. Coppersmith, S. Winograd "Matrix Multiplication via Arithmetic Progressions", *Proc. 19th Symp. on the Theory of Computing*, pp. 1-7, New York, 1987.

[Pe] D. Pearce "Rationalizable Strategic Behavior and the Problem of Perfection", *Econometrica*, 52, 1029-1050, 1985.

[1] As we discussed in the beginning of Section 2, there may be many possible outcomes of the strategy elimination process. However, it follows from our construction of matrix A that whether one of the first m rows or n columns is deleted does not depend on the order of the eliminations.