

Parallel Computation of Nash Equilibria in N-Player Games

Jonathan Widger
Department of Computer Science
Wayne State University
5143 Cass Avenue
Detroit MI 48202, USA
Email: dw0587@wayne.edu

Daniel Grosu
Department of Computer Science
Wayne State University
5143 Cass Avenue
Detroit MI 48202, USA
Email: dgrosu@cs.wayne.edu

Abstract—We propose a parallel algorithm for finding Nash equilibria in n -player noncooperative games. The algorithm is based on enumerating the supports of mixed strategies in parallel and solving for each support the corresponding multilinear equations giving the candidate equilibria. We implemented the proposed algorithm using MPI on a cluster of computers. We performed extensive experiments on random games to show the performance of the parallel algorithm.

Keywords-game theory; Nash equilibrium; parallel algorithm;

I. INTRODUCTION

Game theory is widely used to model the interactions between rational agents and has profound importance in economics. It has also impacted other fields such as biology, political sciences, philosophy, and more recently, computer science. Applications of game theory to computer science can be found in networking, electronic commerce, multi-agent systems, sponsored search advertisements, and resource allocation [1].

The strategic behavior of rational, profit maximizing, agents in a noncooperative game is predicted to be a Nash equilibrium, a situation where no player can do any better by unilaterally changing her strategy. Nash proved that this equilibrium concept exists for any game with a finite number of players each having a finite number of strategies [2]. In practice, it is very important to be able to compute Nash equilibria since they represent predictions of the outcome of the interactions among the strategic agents. The problem of computing the Nash equilibria has deep implications in theoretical computer science as it is considered to be the “most important concrete open question at the boundary of P today” [3].

Computing Nash equilibria in small games using sequential algorithms is trivial due to the small problem size. Typically these games consists of two players and only a handful of actions for each player. Real-life situations can rarely be modeled using small size games. These real-life situations require the use of games with many players and many actions, making finding the Nash equilibria quite difficult. In the worst case, current algorithms cannot guarantee better than exponential time to find even one Nash equilibrium [1].

Most algorithms for finding Nash equilibria focus on two-player games, or only on finding one equilibrium point, and have been studied extensively [4], [5], [6]. Our previous work has shown that some of the algorithms for finding equilibria in two-player games have good parallel solutions [7]. In this paper, however, we focus on finding all Nash equilibria of a game with an arbitrary number of players. The most notable algorithm for computing a sample Nash equilibrium in games with more than two players is the Govindan-Wilson algorithm [8]. It works by perturbing the input game into one that has a known equilibrium, and then it works backwards from the perturbed game to the original by reversing the perturbations. Another algorithm for finding a sample Nash equilibrium is based on simplicial subdivision [9], which approximates a fixed point of the best response function defined on a product of simplices. Simple search algorithms for finding a sample Nash equilibrium have been proposed in [4]. Datta [10] investigated different methods used to solve the polynomial equations characterizing the Nash equilibria in n -player games, showing that polyhedral continuation method is the most promising candidate.

The de facto standard software tool for finding Nash equilibria and analyzing games is Gambit [11]. Gambit implements a number of algorithms, including the Govindan-Wilson algorithm. Another software package, GameTracer [12], implements the Govindan-Wilson algorithm for solving n -player games. Due to the complexity of real-life games, they cannot be solved using either of these tools in a reasonable amount time. Short of designing new algorithms to compute Nash equilibria, a way of solving large scale games in a reasonable amount of time is to exploit the inherent parallelism in equilibria computation. No known software tools take advantage of this parallelism in finding Nash equilibria.

A. Our contributions

In this paper, we propose a parallel totally mixed Nash equilibria enumeration algorithm for n -player games. In order to solve the system of multilinear equations characterizing each potential totally mixed Nash equilibrium we use PHCpack [13], a solver for polynomial systems. The

parallel algorithm is implemented using MPI [14]. We test our implementation with a series of experiments on a grid system and analyze the performance of our algorithm.

B. Organization

The paper is structured as follows. In Section II, we present the basic concepts of n -player games and equilibria. In Section III, we describe the sequential totally mixed enumeration algorithm for computing Nash equilibria in n -player games. In Section IV, we present the design of our parallel algorithm. In Section V, we investigate the performance of our parallel algorithm. In Section VI, we draw conclusions and present some ideas for future work.

II. N -PLAYER GAMES AND EQUILIBRIA

A strategic noncooperative game models the interactions among multiple decision-makers. This type of game is defined as follows.

Definition 1. A noncooperative game Γ consists of:

- (i) a set of n players, $N = \{1, \dots, n\}$.
- (ii) for each player i , a finite set of actions, or pure strategies, $S_i = \{s_{i1}, s_{i2}, \dots, s_{ik_i}\}$, where $k_i = |S_i|$.
- (iii) for each player i , a payoff function $U_i : S \rightarrow \mathbb{R}$, where $S = S_1 \times S_2 \times \dots \times S_n$.

In a strategic game of complete information the players know the structure of the game and choose their strategies simultaneously. Player i 's payoff function characterizes her preferences over the set of strategy profiles $s \in S$. In several games the payoff function is given in the form of a matrix of real values. One type of payoff matrix representation is shown in the following example.

Example 1. As an example we consider a normal form game with three players and two actions per player. The payoffs are given by the following matrix.

$$X = \begin{bmatrix} 9 & 4 & 6 & 3 & 1 & 7 & 2 & 8 \\ 1 & 5 & 2 & 4 & 3 & 8 & 2 & 9 \\ 7 & 1 & 7 & 2 & 6 & 3 & 5 & 1 \end{bmatrix}$$

In this game representation each player's payoff values are given in a particular row, that is player 1's payoffs are given in the first row, player 2's payoffs in the second row, and so on. Each column of this matrix corresponds to a strategy profile. The first column corresponds to the strategy profile (s_{11}, s_{21}, s_{31}) , composed of the first strategies of the three players. The second column corresponds to strategy profile (s_{11}, s_{21}, s_{32}) , which is player 1 choosing her first strategy, player 2 choosing her first strategy, and player three choosing her second strategy. The last column corresponds to strategy profile (s_{12}, s_{22}, s_{32}) , which is all players choosing their second strategies. In the case of strategy profile (s_{11}, s_{21}, s_{32}) , the payoffs to the three players are $X(1, (s_{11}, s_{21}, s_{32})) = 4$, $X(2, (s_{11}, s_{21}, s_{32})) = 5$, and $X(3, (s_{11}, s_{21}, s_{32})) = 1$. In this notation, the first index

represents the player number, while the second represents the strategy profile.

In order to simplify the description, in the rest of the paper we use the following notation. For a strategy profile $s \in S$ we denote by $s_i \in S_i$ the strategy played by player i , and by s_{-i} the strategies played by the other players. We denote the payoff of player i obtained at strategy profile $s \in S$ by $U_i(s)$.

The Nash equilibrium is the most widely used solution concept for noncooperative games. It is defined as follows.

Definition 2. A Nash equilibrium for a game Γ is a strategy profile $s = (s_1, \dots, s_n)$, with the property that for every player i , and for all $s'_i \in S_i$, we have,

$$U_i(s_i, s_{-i}) \geq U_i(s'_i, s_{-i})$$

In other words, at the Nash equilibrium no player can improve upon her payoff by changing her own strategy given that all other players play their equilibrium strategies.

A mixed strategy of player i , denoted by σ_i , is a probability distribution over player i 's pure strategies S_i , i.e., $\sigma_i = (\sigma_{i1}, \dots, \sigma_{ik_i})$ where σ_{ij} is the probability of choosing pure strategy s_{ij} . For each player i , σ_{ij} must satisfy $0 \leq \sigma_{ij} \leq 1$ for each pure strategy $s_{ij} \in S_i$, and $\sum_{j \in \{1, \dots, k_i\}} \sigma_{ij} = 1$. A mixed strategy profile for the game is then $\sigma = (\sigma_1, \dots, \sigma_n)$. The mixed strategy profile of the players other than i is denoted by $\sigma_{-i} = (\sigma_1, \dots, \sigma_{i-1}, \sigma_{i+1}, \dots, \sigma_{ik_i})$. We denote by Σ_i the set of mixed strategies of player i and by Σ the set of mixed strategy profiles. Since now we are dealing with mixed strategies, the preference of the players is characterized by expected payoffs. We denote the expected payoff of player i by $\bar{U}_i(\sigma)$.

Definition 3. A mixed strategy Nash equilibrium for a game Γ is a mixed strategy profile $\sigma = (\sigma_1, \dots, \sigma_n)$, with the property that for every player i , and for all $\sigma'_i \in \Sigma_i$, we have,

$$\bar{U}_i(\sigma_i, \sigma_{-i}) \geq \bar{U}_i(\sigma'_i, \sigma_{-i})$$

Similar to the definition of Nash equilibrium in pure strategies, this states that keeping all other player's mixed strategies fixed, a player does not have incentives to deviate from his or her own mixed strategy. Nash [15] proved that any game with a finite set of players each having a finite set of pure strategies has a mixed Nash equilibrium. Finding mixed strategies that are best responses to each other reduces to solving a linear complementarity problem [16].

Definition 4. A Nash equilibrium $\sigma = (\sigma_1, \dots, \sigma_n)$ is called totally mixed if for each player i , every pure strategy s_{ij} , $j \in \{1, \dots, k_i\}$ occurs with positive probability in σ_i .

A totally mixed Nash equilibrium occurs when for each player i , keeping the mixed strategy profiles for all other players, σ_{-i} , fixed, the payoffs from player i 's pure strategies are equal. This satisfies Definition 3, that is, player i

cannot improve upon his or her own payoff by any unilateral change in strategy. If the payoffs of one or more pure strategies are higher than others, the player may choose to set the probability of playing pure strategies with lesser payoffs to zero in order to maximize her expected payoff for the game, thus, conflicting with the Nash equilibrium conditions. This suggests a procedure for computing Nash equilibria which is presented in the next section.

III. SEQUENTIAL ALGORITHM

The Nash equilibria of an n -player game can be computed by enumerating all possible totally mixed strategy profiles and checking if a Nash equilibrium exists. Thus, the algorithm must generate all possible combinations of pure strategies for each player and generate all possible combinations of these combinations to represent totally mixed strategy profiles. We denote by ρ_{ij} the expected payoff of player i when she plays strategy $s_{ij} \in S_i$ and the other players play according to the mixed strategy profile σ_{-i} . This payoff can then be calculated from the following multilinear form:

$$\rho_{ij} = \bar{U}_i(s_{ij}, \sigma_{-i}) = \sum_{j_1=1}^{k_1} \cdots \sum_{j_{i-1}=1}^{k_{i-1}} \sum_{j_{i+1}=1}^{k_{i+1}} \cdots \sum_{j_n=1}^{k_n} U_i(s_{1j_1}, \dots, s_{(i-1)j_{i-1}}, s_{ij}, s_{(i+1)j_{i+1}}, \dots, s_{nj_n}) \quad (1)$$

$$\sigma_{1j_1} \cdots \sigma_{(i-1)j_{i-1}} \sigma_{(i+1)j_{i+1}} \cdots \sigma_{nj_n}$$

A candidate solution for a Nash equilibrium is determined by checking if the equilibrium conditions can be satisfied. That is, for each player i , one could not improve upon his or her own expected payoff by keeping the strategies of the other players constant and deviating with his or her own. For each player i one has to construct and solve the following equations for each pure strategy s_{ij} that has positive probability in σ_i :

$$\begin{aligned} \rho_{i1} &= w_i \\ \rho_{i2} &= w_i \\ \rho_{i3} &= w_i \\ &\dots \\ \rho_{ij} &= w_i \end{aligned} \quad (2)$$

where w_i is the expected payoff obtained by i . The above system of multilinear equations says that the expected payoff for player i 's first pure strategy represented in σ_i must be equal to the expected payoff of all her other pure strategies represented in σ_i . By transitivity, this says that all of the expected payoffs for pure strategies in σ_i are equal. If they were not equal, player i could drop the pure strategy that broke the equality and improve upon his or her total expected payoff.

For each player i we denote by $\text{support}(\sigma_i)$ the support of mixed strategy σ_i and by Σ_i the set of completely mixed

strategies. Due to being multilinear, the set of equations may have more than one solution. Each of these must be checked for three conditions in order to determine if it represents a Nash equilibrium. First, to satisfy the definition of a totally mixed equilibrium, the probabilities in each solution must sum to 1 and must be strictly positive. That is:

$$\sum_{j=1}^{k_i} \sigma_{ij} = 1 \quad (3)$$

and

$$\text{for all } s_{ij} \in \text{support}(\sigma_i), \sigma_{ij} > 0 \quad (4)$$

Next, the solutions of the system of multilinear equations may be complex numbers. Because we limit ourselves to real solutions we must determine whether the imaginary coefficient ι_{ij} is non-zero for each player i and each pure strategy s_{ij} represented in σ_i , that is we check the following condition:

$$\text{for all } s_{ij} \in \text{support}(\sigma_i), \iota_{ij} = 0 \quad (5)$$

Finally, we must verify that the expected payoff for each pure strategy represented in the totally mixed equilibrium is maximal and equal to each other. For each player i and each pure strategy s_{ij} represented in σ_i and pure strategies s_{il} not represented in σ_i we check the following conditions:

$$\text{for all } s_{ij} \in \text{support}(\sigma_i), \rho_{i1} = \rho_{ij} \quad (6)$$

$$\text{for all } s_{il} \in S_i, \text{ and, } s_{il} \notin \text{support}(\sigma_i), \rho_{i1} \geq \rho_{il} \quad (7)$$

Example 2. We consider the same three-player game with two actions for each player presented in Example 1. There is only one mixed strategy profile to test in this case. In order to conform with the definition of mixed strategy Nash equilibria we have to show that by keeping two player's strategies fixed, the third player does not have incentives to change her own strategy. In other words, if the third player can neither improve nor reduce her expected payoffs, the Nash equilibrium conditions hold for that player's mixed strategy. Therefore, we need to show that the strategies a player has, have indifferent, thus equal, expected payoff values. For player 1, this can be accomplished by solving

$$\bar{U}_1(s_{11}, \sigma_2, \sigma_3) = \bar{U}_1(s_{12}, \sigma_2, \sigma_3)$$

which is

$$\rho_{11} = \rho_{12}$$

This reduces to solving,

$$\sum_{j=1}^2 \sum_{k=1}^2 (X(1, (s_{11}, s_{2j}, s_{3k})) - X(1, (s_{12}, s_{2j}, s_{3k}))) \sigma_{2j} \sigma_{3k} = 0$$

Writing these equations for each player and replacing the payoff values with the ones given by payoff matrix X we

obtain the following system of equations.

$$\begin{aligned}
& 9\sigma_{21}\sigma_{31} + 6\sigma_{22}\sigma_{31} + 1\sigma_{21}\sigma_{32} + 2\sigma_{22}\sigma_{32} \\
& - (4\sigma_{21}\sigma_{31} + 3\sigma_{22}\sigma_{31} + 7\sigma_{21}\sigma_{32} + 8\sigma_{22}\sigma_{32}) = 0 \\
& \sigma_{11}1\sigma_{31} + \sigma_{12}5\sigma_{31} + \sigma_{11}3\sigma_{32} + \sigma_{12}8\sigma_{32} \\
& - (\sigma_{11}2\sigma_{31} + \sigma_{12}4\sigma_{31} + \sigma_{11}2\sigma_{32} + \sigma_{12}9\sigma_{32}) = 0 \\
& \sigma_{11}\sigma_{21}7 + \sigma_{12}\sigma_{21}1 + \sigma_{11}\sigma_{22}7 + \sigma_{12}\sigma_{22}2 \\
& - (\sigma_{11}\sigma_{21}6 + \sigma_{12}\sigma_{21}3 + \sigma_{11}\sigma_{22}5 + \sigma_{12}\sigma_{22}1) = 0
\end{aligned}$$

Since $\sigma_{12} = 1 - \sigma_{11}$, $\sigma_{22} = 1 - \sigma_{21}$, and $\sigma_{32} = 1 - \sigma_{31}$, we can reduce the number of unknowns to three, σ_{11} , σ_{21} and σ_{31} , and solve the system. We find two solutions,

$$\sigma_{11} = \frac{1}{2}, \sigma_{21} = \frac{3}{4}, \sigma_{31} = \frac{4}{7}$$

and,

$$\sigma_{11} = \frac{7}{8}, \sigma_{21} = \frac{3}{2}, \sigma_{31} = \frac{1}{2}$$

Since each of the probabilities in the first solution are positive and less than 1, this strategy profile constitutes a totally mixed Nash equilibrium. The second solution to the set of multilinear equations does not satisfy the definition of a mixed strategy, and therefore cannot be a Nash equilibrium. Since there is only one solution we do not need to check whether it offers the maximal payoff over each player's pure strategies.

The sequential algorithm for finding all Nash equilibria by enumerating totally mixed equilibria is given in Algorithm 1. Before the algorithm is executed, the data is loaded in the payoff matrix X . Based on the game data, the values for the number of players and number of actions for each player are determined. The algorithm enumerates the possible combinations of pure strategies of any size for each mixed strategy for all players. In line 2 we solve the system of multilinear equations given in equation (2), for all players, using PHCpack [13]. The Nash equilibrium conditions for each player are checked in lines 3 to 27. The expected payoff ρ_{ij} is calculated considering the solutions of the multilinear system obtained in line 2. Line 10 checks if equations (4), (5), and (6) are satisfied. In other words, for each support of the mixed strategy it checks that the probability in the support is non-zero and less than one, the probability's imaginary coefficient is zero, and the expected payoffs for the pure strategies in the support are equal. Lines 13 through 16 determine whether the condition in equation (7) holds; the expected payoffs for the pure strategies in the support are equal to or greater than the expected payoffs for all pure strategies not in the support for that player. Lines 18 through 21 check if equation (3) holds, that is the probabilities defining the mixed strategy sum to 1. Finally, lines 24 through 26 say that if the Nash conditions have been satisfied then print the details of the mixed Nash equilibrium found.

Algorithm 1 Sequential Totally Mixed Enumeration

```

1: for all supports of  $\sigma = (\sigma_1, \dots, \sigma_n), \sigma_1 \in \Sigma_1, \sigma_2 \in \Sigma_2, \dots, \sigma_n \in \Sigma_n$  do
2:    $\sigma \leftarrow \text{PHC}(X, \sigma)$   $\triangleright$  solve the system in eq. (2)
3:   for all  $i \in \{1, \dots, n\}$  do
4:     for all  $j \in \{1, \dots, k_i\}$  do
5:       compute  $\rho_{ij}$  using eq. (1)
6:     end for
7:      $psum \leftarrow 0$ 
8:      $nash \leftarrow \text{true}$ 
9:     for all  $s_{ij} \in \text{support}(\sigma_i)$  do
10:      if  $(\sigma_{ij} > 1)$  or  $(\sigma_{ij} \leq 0)$  or  $(\iota_{ij} \neq 0)$  then
11:         $nash \leftarrow \text{false}$ 
12:      end if
13:      for all  $s_{il} \in S_i - \text{support}(\sigma_i)$  do
14:        if  $\rho_{il} \geq \rho_{ij}$  then
15:           $nash \leftarrow \text{false}$ 
16:        end if
17:      end for
18:       $psum \leftarrow psum + \sigma_{ij}$ 
19:    end for
20:    if  $psum \neq 1$  then
21:       $nash \leftarrow \text{false}$ 
22:    end if
23:  end for
24:  if  $nash$  is true then
25:    print( $\sigma$ )
26:  end if
27: end for

```

The complexity of generating all the supports in the sequential totally mixed enumeration algorithm for each m -sized subset of pure strategies for each player is $O\binom{k_i!}{m!(k_i-m)!}$. Using Stirling's approximation we can estimate this to $O(k_i^{k_i})$. If all of the players have the same number of pure strategies, k , then the overall growth can be estimated at $O(nk^{k+1})$.

Each combination of pure strategies generates a system of multilinear equations which we solve using PHCpack [13]. PHCpack employs homotopy continuation to approximate all solutions of a system of n equations with n unknowns. It accomplishes this as follows. First, it exploits the algebraic structure of the input system to count the number of roots and to make a start system whose solution is very easy to compute or already known. The number of roots is central to homotopy continuation as it determines the number of paths that must be explored to find unique solutions. It is also used in validating numerical results. Then, the continuation methods use the start system as a model for finding the desired solutions for the input system by iteratively perturbing the system slightly towards the coefficients of the input.

Computing one solution to a multilinear system of equa-

tions is NP-Hard. An oracle can tell us which roots solve the system and in polynomial time we can verify those results. Harder is computing all solutions to a given input system. There does not exist a polynomial algorithm to verify that a set of solutions corresponds to correct results. As such, complexity must be measured in terms of the number of solutions of a given system, however this is only known when the system is solved [17]. Thus, solving the system of equations is #P-Hard [18].

IV. PARALLEL ALGORITHM

The sequential totally mixed enumeration algorithm exhibits very good potential for parallelism because each totally mixed strategy profiles can be processed independently. Once a unique totally mixed strategy profile is enumerated, the computation can be performed on an independent node of a parallel machine. The cost of enumerating the mixed equilibria is dwarfed by the cost of calling PHCpack and performing the verification process to see if the Nash equilibrium conditions are satisfied.

We consider a message-passing system consisting of P processors. At the start of execution, each processor is assigned a unique processor identification number, $p_{id} \in \{0 \dots P\}$. To avoid any costly communication each processor loads the payoff matrix into its own memory. Then, each processor begins enumerating candidate totally mixed profiles, but only processes a fraction of them based on P . Using an internal counter that increments after each enumeration, $count$, a processor only performs computations on enumerations where $count \bmod P = p_{id}$. This way, using a round robin distribution, each processor is assigned a relatively equal number of totally mixed strategy profiles to verify.

The parallel totally mixed enumeration algorithm is given in Algorithm 2. The majority of the parallel algorithm is the same as the sequential algorithm. The only differences can be seen on line 1, where the counter is initialized, and lines 3 and 4, where the round robin check is made. All of the Nash equilibrium criteria checks are the same, as well as calling PHCpack to solve the system of equations.

Due to the complete lack of dependencies between tasks or communication between processors we expect the algorithm to achieve a speedup close to P . Any communication, waiting for an order of execution, or non-parallelizable sections reduce a parallel algorithm's overall parallel performance. Of these items, the portion of the algorithm that performs the enumeration remains sequential. However, it is not expected that this will greatly impact the overall performance. Our experimental results reflect these expectations.

V. EXPERIMENTAL RESULTS

We ran our experiments on two symmetric multiprocessor machines each running 8 Intel Xeon cores at 2.5 GHz and 8GB of DDR2 memory and a 1Gb/s Ethernet network

Algorithm 2 Parallel Totally Mixed Enumeration

```

1:  $count \leftarrow 0$ 
2: for all supports of  $\sigma = (\sigma_1, \dots, \sigma_n), \sigma_1 \in \Sigma_1, \sigma_2 \in \Sigma_2, \dots, \sigma_n \in \Sigma_n$  do
3:    $count \leftarrow count + 1$ 
4:   if  $count \bmod P = p_{id}$  then
5:      $\sigma \leftarrow \text{PHC}(X, \sigma) \triangleright$  solve the system in eq. (2)
6:     for all  $i \in \{1, \dots, n\}$  do
7:       for all  $j \in \{1, \dots, k_i\}$  do
8:         compute  $\rho_{ij}$  using eq. (1)
9:       end for
10:       $psum \leftarrow 0$ 
11:       $nash \leftarrow \text{true}$ 
12:      for all  $s_{ij} \in \text{support}(\sigma_i)$  do
13:        if  $(\sigma_{ij} > 1)$  or  $(\sigma_{ij} \leq 0)$  or  $(t_{ij} \neq 0)$  then
14:           $nash \leftarrow \text{false}$ 
15:        end if
16:        for all  $s_{il} \in S_i - \text{support}(\sigma_i)$  do
17:          if  $\rho_{il} \geq \rho_{ij}$  then
18:             $nash \leftarrow \text{false}$ 
19:          end if
20:        end for
21:         $psum \leftarrow psum + \sigma_{ij}$ 
22:      end for
23:      if  $psum \neq 1$  then
24:         $nash \leftarrow \text{false}$ 
25:      end if
26:    end for
27:    if  $nash$  is true then
28:       $\text{print}(\sigma)$ 
29:    end if
30:  end if
31: end for

```

link between them. As a programming platform we used MPICH [14], a portable implementation of the Message-Passing Interface standard.

The test games were generated using GAMUT [19], a de facto standard utility for generating games. GAMUT does not provide many types of games that support more than two players. We chose to test with randomly generated games due to their flexibility. In our tests each player has the same number of pure strategies as every other player, however the algorithm does support an arbitrary number of pure strategies for each player. For simplicity we chose games with integer payoff values. The GAMUT command line arguments used to create the games were: -g RandomGame -int_payoffs -players n -actions k , where $n = 2, \dots, 7$ is the number of players, and $k = 2, \dots, 7$ is the number of pure strategies for each player. The combinations of players and pure strategies were constrained by the maximum number of totally mixed equilibria supported by an n - k game [20]. Due to this constraint we were unable to test games with seven

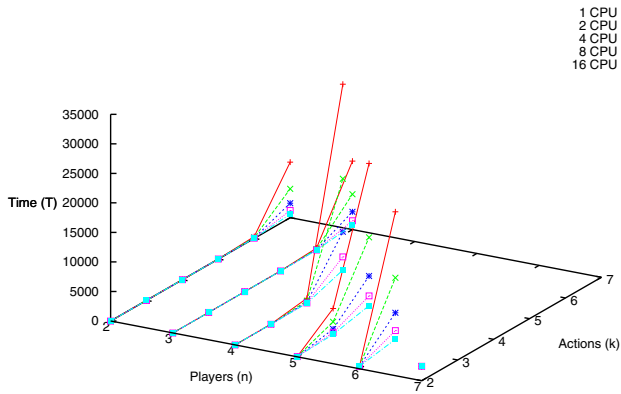


Figure 1. Execution time vs. number of players and actions

players and more than two pure strategies; the figures show results from these games as only a set of points without lines. Three games were generated for each combination of players and pure strategies and the results were averaged.

The execution time for each game grows exponentially with both the number of players and the number of strategies, as previously predicted. Figure 1 shows that the execution time when using only one processor increases sharply for small games taking seconds to solve to games that take hours to solve. As an example, we can observe the growth from a game with 4 players and 4 actions which took on average 898.022 seconds to complete, to a game with 4 players and 5 actions requiring 33651.328 seconds. We can see similar growth by keeping the number of actions constant and increasing the number of players. A game with 5 players and 4 actions took on average 25744.177 seconds to complete on a system with only one processor. The growth in the execution time is also due to the process used to solve the multilinear system of equations. The game with 6 players and 2 pure strategies took 395.371 seconds to complete, however the game with 7 players and 2 pure strategies took 2368.777 seconds. Both of these games have only one system to solve.

Utilizing multiple processors we saw a dramatic decrease in the amount of time it took to solve a game in almost all cases. Games that took hours to complete using the sequential algorithm may only take minutes using the parallel algorithm due to the ease of parallelization, a minimal amount of non-parallelizable regions, and no task dependencies between processors. The exception to this finding is when a game has two pure strategies for each player. In that case there is only one combination of mixed strategies in the game, and therefore only one processor does the work. In that case the sequential algorithm is faster than the parallel algorithm due to the small overhead incurred in setting up

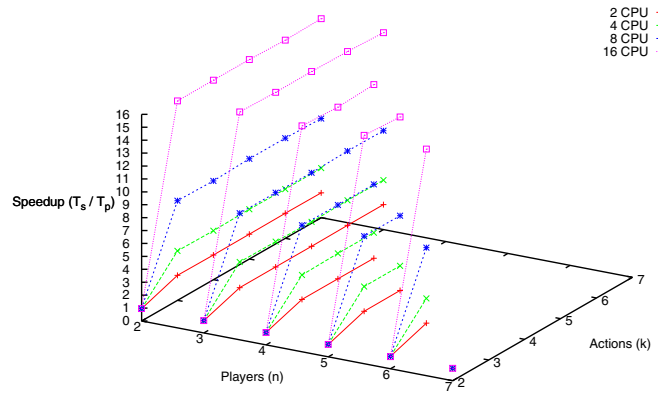


Figure 2. Speedup vs. number of players and actions

the message passing communication links. In all other cases, the parallel algorithm is much faster.

In Figure 2, we compute the speedup of the parallel algorithm versus the sequential algorithm for games of varying size. Speedup S is defined as the ratio of the serial execution time T_s and the parallel execution time T_p , $S = \frac{T_s}{T_p}$. An ideal speedup is obtained when this ratio is equal to the number of processors used to execute the parallel algorithm. We can see that the speedup quickly rises to the number of processors used in the parallel execution tests. Again the exception is when there are only two pure strategies for each player and only one processor is being used throughout the entire execution of those particular games. In a more typical case, such as 3 players and 6 pure strategies, we can see the average speedup to be 1.936 when using 2 processors, which is very close to 2. We can see that when using 16 processors on the same game we have an average speedup of 15.381. For nearly any game we can expect that the speedup is going to be close to the number of processors used in the parallel machine when using this algorithm.

Figure 3 shows how efficiently the processors are used. Efficiency, E , is the ratio of the speedup and the number of processors used, $E = \frac{S}{P}$. An ideal efficiency is 1, saying that the sequential algorithm was parallelized efficiently over all the processors. Since some portions of any algorithm are inherently sequential the objective is to obtain an efficiency as close to 1 as possible. We see in Figure 3 that except for the case of two pure strategies for each player the efficiency is close to 1. For all games that we tested the efficiency was over 95% if each player had 3 or more pure strategies. This tells us that the overhead for generating the combinations of pure strategies for each totally mixed equilibrium is small. Also, the distribution technique to guarantee that each processor has a similar amount of workload as any other induces very low overhead. Thus, we expect good efficiency

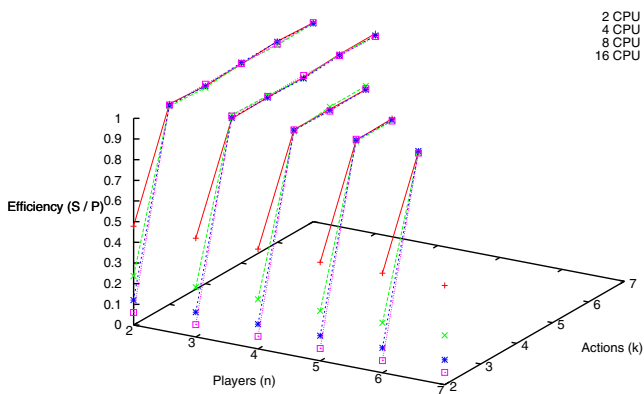


Figure 3. Efficiency vs. number of players and actions

when both larger games and larger parallel machines are considered.

VI. CONCLUSION

We designed, implemented, and tested a parallel algorithm for finding all Nash equilibria in an n -player game through enumerating the set of all possible supports of mixed strategies. The performance of the algorithm was analyzed for games of different combinations of players and pure strategies for each player. The analysis shows that the algorithm scales very well and could potentially be used for much larger games and on much larger systems than the current testbed used in our experiments.

Future work includes creating a portable parallel software package for finding Nash equilibria including this algorithm and others. Further improvements can be made to allow near-optimal allocations of work in parallel machines with heterogeneous nodes.

ACKNOWLEDGMENT

This research was supported in part by NSF grant DGE-0654014.

REFERENCES

- [1] N. Nisan, T. Roughgarden, E. Tardos, and V. Vazirani, *Algorithmic Game Theory*. New York: Cambridge University Press, 2007.
- [2] J. F. Nash, "Equilibrium points in n -person games," *Proc. Nat'l Academy of Sciences of the United States of Am.*, vol. 36, no. 1, pp. 48–49, January 1950.
- [3] C. Papadimitriou, "Algorithms, Games and the Internet," in *Proceedings of the ACM Symposium on Theory of Computing*, July 2001, pp. 749–753.
- [4] R. Porter, E. Nudelman, and Y. Shoham, "Simple Search Methods for Finding a Nash Equilibrium," in *Proc. of the 19th National Conference on Artificial Intelligence*, July 2004, pp. 664–669.
- [5] T. Sandholm, A. Gilpin, and V. Conitzer, "Mixed-Integer Programming Methods for Finding Nash Equilibria," in *Proceedings of the 20th National Conference on Artificial Intelligence*, July 2005, pp. 495–501.
- [6] R. Savani and B. von Stengel, "Hard-to-Solve Bimatrix Games," *Econometrica*, vol. 74, no. 2, pp. 397–429, 2006.
- [7] J. Widger and D. Grosu, "Computing Equilibria in Bimatrix Games by Parallel Support Enumeration," in *Proc. of the 7th International Symposium on Parallel and Distributed Computing*. IEEE Computer Society Press, July 2008, pp. 250–256.
- [8] S. Govindan and R. Wilson, "A Global Newton Method to Compute Nash Equilibria," *Journal of Economic Theory*, vol. 110, pp. 65–86, 2003.
- [9] G. van der Laan, A. J. J. Talman, and L. van der Heyden, "Simplicial Variable Dimension Algorithms for Solving the Nonlinear Complementarity Problem on a Product of Unit Simplices using a General Labelling," *Mathematics of Operations Research*, vol. 12, no. 3, pp. 377–397, 1987.
- [10] R. S. Datta, "Using Computer Algebra to Find Nash Equilibria," in *Proc. of the International Symposium on Symbolic and Algebraic Computation*, August 2003, pp. 74–79.
- [11] R. D. McKelvey, A. McLennan, and T. Turocy, "Gambit: Software tools for game theory, Version 0.2007.01.30," Available at <http://gambit.sourceforge.net>, 2007.
- [12] B. Blum, D. Koller, and C. Shelton, "Game theory: Gametracer," <http://dags.stanford.edu/Games/gametracer.html>.
- [13] J. Verschelde, "PHCpack: a general-purpose solver for polynomial systems by homotopy continuation," Available at <http://www.math.uic.edu/jan/PHCpack/phcpack.html>, 1999.
- [14] "MPICH - A Portable Implementation of MPI," <http://www-unix.mcs.anl.gov/mpi/mpich1/>.
- [15] J. F. Nash, "Non-cooperative games," *Annals of Math.*, vol. 54, no. 2, pp. 286–295, 1951.
- [16] R. Cottle, J.-S. Pang, and R. E. Stone, *The Linear Complementarity Problem*. San Diego, California: Academic Press, 1992.
- [17] L. Blum, F. Cucker, M. Shub, and S. Smale, *Complexity and Real Computation*. Springer-Verlag, 1997.
- [18] M. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W.H. Freeman and Company, 1979.
- [19] E. Nudelman, J. Wortman, Y. Shoham, and K. Leyton-Brown, "Run the GAMUT: A Comprehensive Approach to Evaluating Game-Theoretic Algorithms," in *Proc. of 3rd International Joint Conference on Autonomous Agents and Multiagent Systems*, August 2004, pp. 880–887.
- [20] R. D. McKelvey and A. McLennan, "The maximal number of regular totally mixed nash equilibria," *Journal of Economic Theory*, vol. 72, pp. 411–425, 1997.