A Technique for Reducing Normal Form Games to Compute a Nash Equilibrium

Vincent Conitzer and Tuomas Sandholm
Carnegie Mellon University, Computer Science Department
5000 Forbes Avenue, Pittsburgh, PA 15213, USA
{conitzer, sandholm}@cs.cmu.edu

Abstract

We present a technique for reducing a normal-form game, $O$, to a smaller normal-form game, $R$, for the purpose of computing a Nash equilibrium. This is done by computing a Nash equilibrium for a subgame, $G$, of $O$ for which a certain condition holds. We also show that such a subgame $G$ on which to apply the technique can be found in polynomial time (if it exists). We show that the technique does not extend to computing Pareto-optimal or welfare-maximizing equilibria. Finally, we present a class of games, which we call ALAGIU (Any Lower Action Gives Identical Utility) games, for which recursive application of (special cases of) the technique is sufficient for finding a Nash equilibrium in linear time.

1 Introduction

Nash equilibrium is the most important solution concept for games. A vector of mixed strategies for the players (that is, for each player, a probability distribution over that player’s actions in the game) is a Nash equilibrium if no individual player can benefit from changing her strategy. Every finite game has at least one Nash equilibrium [10]. However, for the concept to be operational, the mere existence result is not sufficient: it needs to be accompanied by an algorithm for finding an equilibrium. Unfortunately, even in 2-player normal-form games, it is still unknown whether a polynomial-time algorithm exists for computing a Nash equilibrium. The Lemke-Howson algorithm [7] is the best-known algorithm for computing a Nash equilibrium in a 2-player normal-form game, but it can require exponential time [14]. Some special cases can be solved in polynomial time (for example, zero-sum games [9]); other related questions, such as finding a welfare-maximizing equilibrium, are NP-hard [4, 3].

Recently, there has been a renewed surge of interest in the computation of Nash equilibria. Christos Papadimitriou has called the basic problem “a most fundamental computational problem whose complexity is wide open” and “together with factoring, [...] the most important concrete open question on the boundary of P today” [12], and new (exponential-time) algorithms have been suggested that search over the supports of the mixed
strategies [13]. Also, there has been growing interest in computing equilibria of games with special structure that allows them to be represented concisely [6, 8, 2, 5, 1].

The idea we pursue in this paper is the following. For the basic problem of computing a Nash equilibrium of a normal-form game, it would be helpful to have a recursive technique that decomposes a Nash equilibrium computation problem into one or more smaller such problems, in such a way that a solution to the original problem can easily be computed from the solutions to the smaller problems. If there were such a technique that could be applied to any game, and that decomposed it into small enough subproblems, then repeatedly applying this technique would constitute a polynomial-time algorithm for computing a Nash equilibrium. However, even if the technique could not be applied to all games, it would still be of interest. It could be used as a preprocessing step in computing a Nash equilibrium, thereby reducing the load on the algorithm that is eventually called to solve the irreducible subproblems. The technique could also be applied at intermediate stages of any other algorithm that works by reducing the size of the game. Moreover, there may be special subclasses of games such that the technique can be applied on any of those games, as well as on any subproblem resulting from applying the technique on those games. In that case, the technique would constitute a polynomial-time Nash equilibrium finding algorithm for the particular subclass of games.

In this paper, we introduce such a technique for 2-player normal-form games. When possible, this technique finds a subgame, $G$, of the original game, $O$, for which a certain condition holds. $G$ is then solved recursively. Based on the recursively computed equilibrium of $G$, the original game $O$ is then reduced to a smaller game, $R$, which is also solved recursively. From the computed equilibria of $G$ and $R$, an equilibrium of the original game $O$ can then easily be constructed. To our knowledge, this is the first recursive technique for computing a Nash equilibrium (other than the iterated elimination of dominated strategies).

## 2 Main technique

In this section, we introduce our main technique. We are given a 2-player normal-form game $O$, in which the row player chooses a pure strategy from $\Sigma_1$ and the column player chooses a pure strategy from $\Sigma_2$. Suppose that the strategies in $\Sigma_1$ can be labeled as $u_i, s_i$, and those in $\Sigma_2$ as $v_j, t_j$, so that the game can be written as follows:
The condition that allows for writing the game like this is the following:

**Condition 1** Against any fixed \( v_j \), all the \( s_i \) give the row player the same utility \( (a_j) \); and against any fixed \( u_i \), all the \( t_j \) give the column player the same utility \( (b_i) \).

Then, let us compute a Nash equilibrium of the game \( G \) (using any technique):
We then compute a Nash equilibrium for $R$ (using any technique), obtaining equilibrium probabilities $p_{Ru_i}, p_{Rs_i}$ for the row player and equilibrium probabilities $p_{Rv_j}, p_{Rt_j}$ for the column player. Then (as we will show later in the section), setting

\[
p_{Ou_i} = p_{Ru_i}, \quad p_{Os_i} = p_{Rs_i}, \quad p_{Ov_j} = p_{Rv_j}, \quad p_{Ot_j} = p_{Rt_j},
\]

constitutes a Nash equilibrium of $O$.

Before proving formally that the technique is correct, we first illustrate it with a small example. Consider the following game $O$:

\[
\begin{array}{ccc}
  & v_1 & t_1 & t_2 \\
 u_1 & 2,2 & 0,3 & 2,3 \\
 s_1 & 1,2 & 4,0 & 0,4 \\
 s_2 & 1,4 & 0,4 & 4,0 \\
\end{array}
\]

The game satisfies Condition 1. Thus, we first solve the subgame $G$:

\[
\begin{array}{ccc}
  & t_1 & t_2 \\
 s_1 & 4,0 & 0,4 \\
 s_2 & 0,4 & 4,0 \\
\end{array}
\]

This is a matching-pennies game where in equilibrium, each player places probability $1/2$ on each pure strategy and receives expected utility $2$.$^{1}$ Thus, the reduced game $R$ becomes:

\[
\begin{array}{c}
  & v_1 & t \\
 u_1 & 2,2 & 1,3 \\
 s & 1,3 & 2,2 \\
\end{array}
\]

Again, this is a matching-pennies game where in equilibrium, each player places probability $1/2$ on each action. Thus, we have discovered an equilibrium for the original game $O$ where $u_1$ and $v_1$ are played with probability $1/2$ each, and $s_1, s_2, t_1, t_2$ are played with probability $(1/2) \cdot (1/2) = 1/4$ each. We now prove the technique is correct in general.

**Theorem 1** Suppose $O$ is reduced to $R$ using the equilibrium $p^G_{u_i}, p^G_{s_i}$ of $G$. If $p^R_{u_i}, p^R_{s_i}, p^R_{v_j}, p^R_{t_j}$ constitute a Nash equilibrium of $R$, then setting $p_{Ou_i} = p^R_{u_i}, p_{Os_i} = p^R_{s_i}, p_{Ov_j} = p^R_{v_j},$ and $p_{Ot_j} = p^R_{t_j}$ constitutes a Nash equilibrium of $O$.

---

$^{1}$We emphasize that there is no restriction that the game $G$ or $R$ should be a zero-sum game (or any other special type of game).
Proof. To show that these mixed strategies constitute a Nash equilibrium for $O$, we first observe that the row player has no incentive to redistribute the probability placed on strategies $s_i$ to another strategy $s_{i'}$. That is, given that the row player plays one of the strategies $s_i$ (note that given this, the probability that the row player plays a given strategy $s_i$ is $p_R^{G} p^G_{s_{i}} / p_R^{G} = p^G_{s_{i}}$), there is no incentive to switch to another strategy $s_{i'}$. The reason is as follows. Given that the column player plays a given strategy $v_j$, both $p^G_{s_{i}}$ and $s_{i'}$ will give the same utility. Given that the column player plays one of the strategies $t_j$ (note that given this, the probability that the column player plays a given strategy $t_j$ is $p_R^{G} p^G_{s_{i}} / p_R^{G} = p^G_{s_{i}}$), $p^G_{s_{i}}$ is a best response because $p^G_{s_{i}}, p^G_{s_{i}'}$ constitute a Nash equilibrium for $G$. As a result, to show that we have an equilibrium for $O$, we do not need to consider deviations to an arbitrary $s_i$; instead, we only need to consider deviations to the mixed strategy $p^G_{s_{i}}$.

Next, we observe that the expected utility for the row player of playing a given $u_i$ is the same as in the equilibrium computed for $R$. The reason is as follows. The probability that the column player plays a given $v_j$ is the same in both $R$ and $O$, and so is the row player’s utility for the outcome $(u_i, v_j)$. Moreover, the probability that the column player plays one of the strategies $t_j$ in $O$ is $p^G_{t_j}$, and the expected utility for the row player of playing $u_i$ in that case is \( \sum_{j=1}^{n} p^G_{t_j} c_{ij} \), which is the same as in $R$.

We also observe that the expected utility for the row player of playing $p^G_{s_{i}}$ (that is, the row player’s mixed strategy given that the row player plays one of the strategies $s_i$) is the same as the expected utility of playing $s$ in the equilibrium computed for $R$. The reason is as follows. The probability that the column player plays a given $v_j$ is the same in both $R$ and $O$, and the row player’s utility in either case is $a_j$. Moreover, the probability that the column player plays one of the strategies $t_j$ in $O$ is $p^G_{t_j}$, and the expected utility for the row player of playing $p^G_{s_{i}}$ in that case is the expected utility the row player gets in the equilibrium of $G$, namely, $\pi^G_{s_{i}}$, which is the same as the row player’s utility for the outcome $(s_i, t_j)$ in $R$.

But from these last two observations, and the fact that we used an equilibrium for $R$, it follows that the row player has no incentive to deviate to any $u_i$ or to $p^G_{s_{i}}$, and from the first observation, the fact that the row player has no incentive to deviate to $p^G_{s_{i}}$ implies that the row player has no incentive to deviate to any $s_i$. Hence, the row player has no incentive to deviate; by symmetry, neither does the column player. \( \blacksquare \)

3 Detecting whether the technique can be applied

It is easy to verify whether Condition 1 holds when the pure strategies of the game are labeled $u_i, s_i, v_j, t_j$. However, in general, this labeling will not be given to us. As an example, suppose we are given the following game:

<table>
<thead>
<tr>
<th></th>
<th>$\beta_1$</th>
<th>$\beta_2$</th>
<th>$\beta_3$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\alpha_1$</td>
<td>4, 0</td>
<td>1, 2</td>
<td>0, 4</td>
</tr>
<tr>
<td>$\alpha_2$</td>
<td>0, 3</td>
<td>2, 2</td>
<td>2, 3</td>
</tr>
<tr>
<td>$\alpha_3$</td>
<td>0, 4</td>
<td>1, 4</td>
<td>4, 0</td>
</tr>
</tbody>
</table>

This game is, in fact, the same game that was used as an example in Section 2, with the rows and columns permuted ($\alpha_1 = s_1, \alpha_2 = u_1, \alpha_3 = s_2, \beta_1 = t_1, \beta_2 = v_1, \beta_3 = t_2$). But we cannot apply the technique without
knowing the labeling of the strategies as \( u_i, s_i, v_j \), and \( t_j \).

Hence, we need an algorithm that finds a labeling of the strategies as \( u_i, s_i, v_j \), and \( t_j \) so that Condition 1 holds. Note that it does not matter whether we label a given strategy (e.g.) \( s_1 \) or \( s_2 \). That is, all we need to do is partition the row player’s strategies \( \Sigma_1 \) into two subsets, the ones labeled \( u_i \) and the ones labeled \( s_i \); and partition the column player’s strategies \( \Sigma_2 \) into two subsets, the ones labeled \( v_j \) and the ones labeled \( t_j \).

There are two trivial ways of labeling the strategies so that the condition holds: 1. Label all strategies \( s_i \) or \( t_j \). 2. Label at most one strategy \( s_i \), and at most one strategy \( t_j \). These trivial labelings are not useful: the first would give us \( G = O \) to recur on, and the second would not reduce the game at all (\( R = O \)). Any other labeling for which Condition 1 is satisfied, though, would give us a game \( G \) that is smaller than \( O \) to recur on, and reduce the game to a game \( R \) that is smaller than \( O \). Moreover, the number of rows (columns) by which \( R \) is smaller than \( O \) is equal to the number of rows (resp. columns) in \( G \), minus 1. Hence, our objective is to find a labeling for which the condition is satisfied, other than the two trivial ones described above.

We now show how to reformulate this problem as a Horn satisfiability problem. (Recall that a satisfiability clause is a Horn clause if it can be written as \((\neg x_1 \lor \neg x_2 \lor \ldots \lor \neg x_n \lor y)\), or equivalently, \( x_1 \land x_2 \land \ldots \land x_n \Rightarrow y \).)

Let the variable \( v(\sigma) \) be \( \text{true} \) if \( \sigma \) is labeled as one of the \( s_i \) or \( t_j \), and \( \text{false} \) if \( \sigma \) is labeled as one of the \( u_i \) or \( v_j \). The key observation is that to satisfy Condition 1, if row player strategies \( \alpha_1 \) and \( \alpha_2 \) obtain different payoffs for the row player against column player strategy \( \beta \), then it cannot be the case that \( \alpha_1 \) and \( \alpha_2 \) are both labeled as one of the \( s_i \) and \( \beta \) is labeled as one of the \( v_j \)—in other words, \( v(\alpha_1) \land v(\alpha_2) \Rightarrow v(\beta) \). Similarly, if column player strategies \( \beta_1 \) and \( \beta_2 \) obtain different payoffs for the column player against row player strategy \( \alpha \), then \( v(\beta_1) \land v(\beta_2) \Rightarrow v(\alpha) \).

Using this, the game described at the beginning of this section has the following Horn clauses: 1. \( v(\alpha_1) \land v(\alpha_2) \Rightarrow v(\beta_1) \land v(\beta_2) \land v(\beta_3) \), 2. \( v(\alpha_1) \land v(\alpha_3) \Rightarrow v(\beta_1) \land v(\beta_3) \), 3. \( v(\alpha_2) \land v(\alpha_3) \Rightarrow v(\beta_2) \land v(\beta_3) \), 4. \( v(\beta_1) \land v(\beta_2) \Rightarrow v(\alpha_1) \land v(\alpha_2) \), 5. \( v(\beta_1) \land v(\beta_3) \Rightarrow v(\alpha_1) \land v(\alpha_3) \), and 6. \( v(\beta_2) \land v(\beta_3) \Rightarrow v(\alpha_2) \land v(\alpha_3) \). It is straightforward to check that the only nontrivial satisfying assignment, that is, the only assignment that A) satisfies all these clauses, B) does not set all variables to \( \text{true} \), and C) sets either at least two \( v(\alpha_i) \) to \( \text{true} \) or at least two \( v(\beta_j) \) to \( \text{true} \); sets \( v(\alpha_1), v(\alpha_3), v(\beta_1), \) and \( v(\beta_3) \) to \( \text{true} \) and everything else to \( \text{false} \). We note that this corresponds to the labeling that we presented. The Horn clauses can be computed efficiently:

**Lemma 1** The set of Horn clauses for a normal-form game can be computed in time \( O(|\Sigma_1|^2; |\Sigma_2| + |\Sigma_1| \cdot |\Sigma_2|^2) \).

We now show that the Horn clauses fully capture the problem, that is, the Horn clauses being satisfied is a necessary and sufficient condition for Condition 1 to be satisfied.

**Theorem 2** A labeling of the strategies as \( u_i, s_i, v_j, \) and \( t_j \) satisfies Condition 1 if and only if it satisfies all the Horn clauses.

**Proof.** First, suppose that not all of the Horn clauses are satisfied. Then (without loss of generality) there exist \( \alpha_{i_1}, \alpha_{i_2}, \) and \( \beta_j \) such that \( v(\alpha_{i_1}) \land v(\alpha_{i_2}) \Rightarrow v(\beta_j), \) \( v(\alpha_{i_1}) \) and \( v(\alpha_{i_2}) \) are set to \( \text{true} \), and \( v(\beta_j) \) is set to \( \text{false} \).

Alternatively the roles of the \( \alpha_i \) and \( \beta_j \) could be reversed, but by symmetry we can without loss of generality
restrict our attention to the first case.) That is, \( \alpha_{i_1} \) and \( \alpha_{i_2} \) are among the strategies \( s_i \), \( \beta_j \) is among the strategies \( v_j \), and \( \alpha_{i_1} \) and \( \alpha_{i_2} \) give the row player different payoffs against \( \beta_j \). But then Condition 1 is not satisfied.

Now, suppose that all of the Horn clauses are satisfied. We must show, for any \( v_j \), that any \( s_{i_1} \) and \( s_{i_2} \) give the row player the same payoff against \( v_j \). (We must show the same for the \( u_i \) and the \( t_j \), but this will follow by symmetry.) Suppose they do not; then one of the Horn clauses must be \( v(s_{i_1}) \land v(s_{i_2}) \Rightarrow v(v_j) \). But this clause would then be false, which is contrary to our assumption.

A general systematic approach to finding a nontrivial satisfying assignment is to start by setting two \( v(\alpha_i) \) (or two \( v(\beta_j) \)) to true, and subsequently iteratively apply the implication clauses; if this process ends without all the variables being set to true, we have found an assignment with the desired properties. If we do this once for every initial pair of \( v(\alpha_i) \) and every initial pair of \( v(\beta_j) \), then we will find an assignment with the desired properties if it exists.

For instance, in the above example, we start by setting \( v(\alpha_1) \) and \( v(\alpha_2) \) to true; then, applying the first implication, we find that all the \( v(\beta_j) \) must be set to true as well, and hence by the last implication so must \( v(\alpha_3) \), so that all variables are set to true. Then, we try again, starting by setting \( v(\alpha_1) \) and \( v(\alpha_3) \) to true; by the second implication, we find that \( v(\beta_1) \) and \( v(\beta_3) \) must also be set to true. Then we apply the fifth implication, but this does not set any new variables to true, so the process ends here, and we have found an assignment with the desired properties.

**Theorem 3** The algorithm described above requires \( O((|\Sigma_1| + |\Sigma_2|)^4) \) applications of a Horn clause.

**Proof.** There are \( O((|\Sigma_1| + |\Sigma_2|)^2) \) iterations of the outer loop (choosing which pair of variables to start with). Moreover, within each iteration, we never need to apply the same clause twice, and there are only \( O((|\Sigma_1| + |\Sigma_2|)^2) \) clauses. ■

### 4 Limitations of the technique

One may wonder whether the technique can be extended to find Nash equilibria with special properties. For example, if we compute a Pareto optimal or social-welfare maximizing Nash equilibrium in each recursive call, will this give us a Pareto optimal or social-welfare maximizing Nash equilibrium for the original game? The following game \( O \) shows that this is not the case.

<table>
<thead>
<tr>
<th></th>
<th>( \alpha_1 )</th>
<th>( \alpha_2 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( v_1 )</td>
<td>1, 1</td>
<td>4, 0</td>
</tr>
<tr>
<td>( u_1 )</td>
<td>0, 4</td>
<td>3, 3</td>
</tr>
<tr>
<td>( s_1 )</td>
<td>0, 0</td>
<td>0, 0</td>
</tr>
<tr>
<td>( s_2 )</td>
<td>0, 0</td>
<td>2, 2</td>
</tr>
</tbody>
</table>

The subgame \( G \) is as follows:

<table>
<thead>
<tr>
<th></th>
<th>( t_1 )</th>
<th>( t_2 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( s_1 )</td>
<td>3, 3</td>
<td>0, 0</td>
</tr>
<tr>
<td>( s_2 )</td>
<td>0, 0</td>
<td>2, 2</td>
</tr>
</tbody>
</table>
It has multiple equilibria, but the only Pareto optimal Nash equilibrium is \( (s_1, t_1) \). Thus the reduced game \( R \) becomes:

<table>
<thead>
<tr>
<th></th>
<th>( v_1 )</th>
<th>( t )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( u_1 )</td>
<td>1, 1</td>
<td>4, 0</td>
</tr>
<tr>
<td>( s )</td>
<td>0, 4</td>
<td>3, 3</td>
</tr>
</tbody>
</table>

This is a Prisoner’s Dilemma game where the only equilibrium is \( (u_1, v_1) \), and thus we will compute the equilibrium \( (u_1, v_1) \) for the original game \( O \). Unfortunately, this is not a Pareto optimal equilibrium for \( O \), because \( (s_2, t_2) \) is also a Nash equilibrium for \( O \).

5 ALAGIU games: A class of games that can be solved by applying the technique recursively

In general, we may only be able to apply the technique a limited number of times, after which we need to resort to other algorithms to obtain Nash equilibria for games in the recursive calls on which the technique cannot be applied. However, in this section, we present a subclass of normal-form games such that the technique works on any of these games, and any game resulting from a recursive call is itself in the subclass as well. Thus, the technique is sufficient for finding a Nash equilibrium.

**Definition 1** We say that a 2-player normal-form game is an ALAGIU (Any Lower Action Gives Identical Utility) game if each player’s strategy set \( \Sigma_i \) is a subset of \( \mathbb{R} \), and the game has the property that for any pure opponent strategy \( \sigma_{-i} \in \Sigma_{-i} \), for any \( \sigma_1^i, \sigma_2^i \in \{ \sigma_i \in \Sigma_i : \sigma_i < \sigma_{-i} \} \), we have \( \pi_i(\sigma_1^i, \sigma_{-i}) = \pi_i(\sigma_2^i, \sigma_{-i}) \). That is, given that a player chooses a lower strategy than the opponent, it does not matter to that player which of the lower strategies she chooses.

Before showing how the technique can be applied to (finite) ALAGIU games, we first give two examples.

**Example 1 (Higher-bidder-wins auctions)** Any two-player single-item auction in which a lower bidder never wins and never pays anything is an ALAGIU game. Note that arbitrary allocation rules are allowed in case of ties. Also, arbitrary payment rules for the winning bidder are allowed, including first price and second price (but also bizarre, e.g. nonmonotonic, payment rules).

**Example 2 (Vendors on a one-way street)** Say we have a hot dog vendor and an ice cream vendor that must each choose a location on a one-way street. Assume that, given that a vendor chooses a location earlier on the street than the other vendor, it does not matter to the earlier vendor where exactly she locates herself (all traffic must pass her anyway). However, it may matter to the later vendor where he locates himself. (For example, it may be bad for the ice cream vendor to immediately follow the hot dog vendor, because potential customers will have their hands full with the hot dog. In contrast, it may be good for the ice cream vendor to locate himself significantly later on the street than the hot dog vendor, to provide some dessert.) Then, this is an ALAGIU game.
Figure 1: Three outcomes for the vendor game. Under the ALAGIU restriction, the hot dog vendor’s utility must be the same for the top two outcomes. However, the ice cream vendor’s utility need not be the same in any of the three outcomes.

The equilibria of ALAGIU games can be complicated. For example, it is possible to construct ALAGIU games of arbitrary size that have only fully mixed equilibria. (We omit the proof due to space constraint.)

It turns out that any finite ALAGIU game takes one of three special forms described below, each of which satisfies Condition 1. Consider the highest strategy in the game, that is, $\sigma_M = \max \{ \sigma : \sigma \in \Sigma_1 \cup \Sigma_2 \}$. If only the row player has $\sigma_M$ in her strategy set ($\sigma_M \in \Sigma_1 - \Sigma_2$), then the game must have the following form:

\[ u_1 = \sigma_M \begin{array}{cccc} t_1 & t_2 & t_3 & \cdots & t_n \\ c_{11}, b_1 & c_{12}, b_1 & c_{13}, b_1 & \cdots & c_{1n}, b_1 \end{array} \]

\[ s_1 \quad s_2 \quad s_3 \quad \vdots \quad s_m \]

On the other hand, if only the column player has $\sigma_M$ in his strategy set ($\sigma_M \in \Sigma_2 - \Sigma_1$), then the game must have the following form:

\[ v_1 = \sigma_M \begin{array}{cccc} t_1 & t_2 & t_3 & \cdots & t_n \\ a_1, d_{11} & a_1, d_{21} & a_1, d_{31} & \cdots & a_1, d_{m1} \end{array} \]

\[ s_1 \quad s_2 \quad s_3 \quad \vdots \quad s_m \]

Finally, if both the row and the column player have $\sigma_M$ in their strategy sets ($\sigma_M \in \Sigma_1 \cap \Sigma_2$), then the game must have the following form:
Our technique can be applied to every one of these three forms. Moreover, in each case the subgame \( G \) is still an ALAGIU game, and thus we can apply the technique recursively. Finally, the reduced game \( R \) is a \( 2 \times 1 \), \( 1 \times 2 \), or \( 2 \times 2 \) game (for the three cases, respectively), so it can easily be solved.

**Theorem 4** Using the technique described in this paper, a Nash equilibrium of a finite ALAGIU game can be computed in time \( O(|\Sigma_1| \cdot |\Sigma_2|) \) (that is, in time linear in the size of the game).

**Proof.** The subgame \( G \) that the algorithm recurs on is one row and/or column smaller than the current game. Whenever \( G \) is one row smaller (which happens \( O(|\Sigma_1|) \) times), after the recursion we need to compute a weighted average of the \( c_{ij} \), which requires \( O(|\Sigma_2|) \) time. Similarly, whenever \( G \) is one column smaller (which happens \( O(|\Sigma_2|) \) times), after the recursion we need to compute a weighted average of the \( d_{ij} \), which requires \( O(|\Sigma_1|) \) time.

\[\blacksquare\]

6 Conclusions

In this paper, we presented a technique for reducing a game, \( O \), to a smaller game, \( R \), for the purpose of computing a Nash equilibrium. This is done by computing a Nash equilibrium for a subgame, \( G \), of \( O \) for which a certain condition holds. We also showed that such a subgame \( G \) on which to apply the technique can be found in polynomial time (if it exists). We showed that the technique does not extend to computing Pareto-optimal or welfare-maximizing equilibria. Finally, we presented a class of games, which we called ALAGIU (Any Lower Action Gives Identical Utility) games, for which recursive application of (special cases of) the technique is sufficient for finding a Nash equilibrium in linear time.

Future research includes extending the techniques presented here to games with more than two players. It also includes testing experimentally whether the technique is helpful in finding equilibria, for instance on test suites of game generators (such as GAMUT [11]).

References


