Chapter 4

Rational Payoff Irrational Equilibria Games

In the previous chapter we presented an algorithm for deciding membership of the classes of games of interest to us. Once it is known that an input game is a member, we proceed to the problem of computing its equilibria. In this chapter we primarily focus on RPIE games and present an algorithm to compute all their equilibria. The correctness of the algorithm and other related results are proved. A detailed example is presented to show the working of the algorithm. We conclude this chapter with a discussion of the complexity of the given algorithm and related issues.

4.1 Underlying Model

RPIE games, by Definition 2.1.4, have totally mixed Nash equilibria. For this reason, we characterize an input RPIE game via a $\mathcal{G}S$ of the form (2.7). Our algorithm for computing all equilibria of the class of RPIE games makes use of Galois groups. The $\mathcal{G}S$ of any RPIE game generates field extensions of $\mathbb{Q}$. Due to this, all the Galois groups in this chapter satisfy Definition 2.3.4.
4.2 Equilibria of RPIE Games

Before formally presenting our algorithm, we briefly discuss the approach and the underlying assumptions. We assume that we have an RPIE game $T$. As described in Section 2.2, we can derive a system of polynomial equations $\mathcal{GS}$ whose solutions include all Nash equilibria of the game $T$. However, some of the solutions of the $\mathcal{GS}$ need not be Nash equilibria; our algorithm rejects these unwanted solutions using different mechanisms. From Bernstein’s Theorem [4] we have an upper bound on the number of solutions a polynomial system can have. Bounds on the number of equilibria have been given in [63, 64]. These bounds constrain the number of solutions to be computed and the number of non-equilibrium solutions to be rejected.

In the initial phase of our method, Buchberger’s algorithm is invoked to derive a univariate polynomial in the Gröbner basis of the $\mathcal{GS}$. \(^1\) Since the game is RPIE, Nash’s theorem [68] guarantees that the univariate polynomial has at least one irrational root. A root of the univariate polynomial is computed and substituted in the triangular form of a Gröbner basis to find a univariate polynomial in some other indeterminate variable. We repeat this procedure at most $\mathcal{K}^+ - n$ times, and at the end have an irrational solution of the $\mathcal{GS}$, a sample solution.

We denote the Galois group of the irreducible part of a univariate polynomials $g_i$ in the Gröbner basis of $\mathcal{GS}$ by $G_i$, $i \in \{1, \ldots, \mathcal{K}^+\}$. By assumption $G_i$’s are known.

In the next phase, we apply the transitive Galois group action corresponding to each indeterminate variable and determine all irrational solutions of the $\mathcal{GS}$, from the orbits of the group action.

The final phase consists of testing all the irrational solutions and rejecting the non-equilibrium solutions. For this we invoke the Nash equilibrium verification algorithm in [30]. An outline of the entire algorithm is presented below:

\(^1\) For further details of Buchberger’s Algorithm see Section A.2 of Appendix A.

\(^2\) Recall that the $\mathcal{GS}$ has finitely many solutions over the complex number field. This finite variety of the $\mathcal{GS}$ (or equivalently zero-dimensional ideal $\mathcal{I}$ of the $\mathcal{GS}$) guarantees a univariate polynomial in its Gröbner basis [8].
4.2. Equilibria of RPIE Games

Algorithm 4.2.1 Computing All Nash Equilibria of an RPIE game.
Input: An RPIE game, Galois groups.
Output: All equilibria of the input RPIE game in set $X$.

1: $\beta = (\beta_1, \beta_2, \ldots, \beta_{K^+})$. \{Initialize an empty tuple to store a sample solution of the $\mathcal{GS}$\}.
2: Construct the $\mathcal{GS}$ of the input game.
3: Call Algorithm 4.2.2 with $\mathcal{GS}$ for computing a sample equilibrium of the input RPIE game.
4: Call the Galois group action Algorithm 4.2.3 with the sample solution tuple saved in $\beta$.
5: Save output of the Algorithm 4.2.3 in $X$.
6: Reject non-equilibrium solutions of the $\mathcal{GS}$ from $X$ using verification algorithm in [30] or criteria (2.6) and (2.2).

Algorithm 4.2.2 Computation of a sample solution with Gröbner basis.
Input: $\mathcal{GS}$ of the input game.
Output: A sample solution $\beta = (\beta_1, \beta_2, \ldots, \beta_{K^+})$ of the input game.

1: With Buchberger’s Algorithm on $\mathcal{GS}$, compute triangular form of Gröbner basis.
2: while one sample solution $\beta$ of the $\mathcal{GS}$ is not constructed do
3: Compute a root $\alpha$ of univariate polynomial – of some indeterminate variable $x_i$ – generated in Step 3.
4: if $\alpha \in \mathbb{Q}$ then
5: Reject $\alpha$ and go to Step 3.
6: else
7: Save $\alpha$ in $\beta$ at location $\beta_i$.
8: end if
9: Substitute the root $\beta_i$ in $\beta$ into $\mathcal{GS}$ and compute a new triangular form with one less indeterminate variable.
10: end while

Following algorithm computes group action by transitive Galois groups. The action is computed for each indeterminate variable $x_i$ by considering it over each coordinate root in the tuple $\beta$. The action generates Galois-conjugates of the roots that are further saved in as solution tuples in the set $X$. 
Algorithm 4.2.3 Computing orbit of a Galois Group Action.

Input: A sample solution $\beta$ of the $\mathcal{GS}$, Galois groups.
Output: All the conjugate solutions of the input sample solution in set $X$.

1: Initialize the processed-elements list $X$ and unprocessed-elements list $U$ as $X = U = \{\beta\}$.
2: while $U$ is not empty do
3:   Let $u = (u_1, u_2, \ldots, u_{K+})$ be the first element of $U$. Delete $u$ from $U$.
4:   for each $i$ and $j$, $g^i_j$ in Galois group $G_i$ and $u_i \in u$. do
5:      Compute the transitive Galois group action $u^g_{i\beta}$.
6:      $\beta' = (u_1^{g_{1}}_{i}, u_2^{g_{2}}_{i}, \ldots, u_{K+}^{g_{K+}}_{i})$.
7:      if $\beta' \notin X$ then
8:         $X = X \cup \{\beta'\}$ and $U = U \cup \{\beta'\}$.
9:      end if
10:   end for
11: end while

Traditional approach, given in [21], for computing solutions of system of polynomial equations using Gröbner basis calls the Buchberger’s algorithm for computing a triangular form. The triangular form provides a univariate polynomial in one indeterminate variable. Each root of the univariate polynomial is then substituted back in the triangular form to compute corresponding solution tuple; the operation requires multiple substitutions and factorizations. Algorithm 4.2.1, on the other hand, invokes Buchberger’s algorithm exactly once. The Algorithm 4.2.2 computes a sample solution tuple corresponding to the first irrational root of the univariate polynomial. Rest of the solutions are then generated by polynomial time group action, requiring no further substitutions and factorizations.

It is important to note that due to Theorem 2.1.3 and the fact that the input game has all irrational equilibria, we are guaranteed to get one solution of the $\mathcal{GS}$ in $\beta$ and so the Algorithm 4.2.1 reaches Step 4, every time. It then calls the Algorithm 4.2.3 for computing polynomial time Galois group action over available sample solution in the $\beta$. In the Algorithm 4.2.3 all other conjugate roots are computed with their known Galois groups $G_i$.

Moreover, finite group action on finite variety guarantees that the Algorithm 4.2.3
reaches Step 11. At the end of Step 11, Algorithm 4.2.3 generates solutions of polynomial system GS in X, all of which may not be Nash equilibria. We use polynomial time algorithm, suggested in [30], to reject the non-equilibrium solutions.

Note that a rational root forces its univariate polynomial to factorize over the field $\mathbb{Q}$. For assuming known Galois group we ignore the linear factor corresponding to each rational root and consider only the irreducible part of the univariate polynomial. We assume that the Galois groups of these irreducible parts are completely known. It also follows from Theorem 2.3.7 that the known Galois groups act transitively on the roots.

**4.2.1 Rational Number Check**

Details of the condition in Step 4, of Algorithm 4.2.2, for deciding $\alpha \in \mathbb{Q}$ are as follows.

Since numbers are stored in computer memory with finite precision, it is a non-trivial task to determine whether a stored number is rational or irrational. The approach that we have adopted for the problem is as follows.

As a first step, a suitable numerical algorithm (polynomial time root approximation methods), is used to compute an approximate root of the univariate polynomial. Given the approximate root, the degree of the univariate polynomial and its height (defined as the Euclidean length of coefficients of the polynomial), it is possible to construct the minimal polynomial for a root. Finally, checking the irreducibility of the minimal polynomial resolves the problem. The minimal polynomial is constructed using the Kannan Lovasz Lenstra (KLL) algorithm [44]. The irreducibility check can be performed using univariate polynomial factorization algorithm over the field of rational numbers [31]. For further details of the KLL algorithm, see Section A.3.

Algorithm 4.2.2 computes a sample solution of the GS. Various approaches for
computing a sample equilibrium of a game are discussed in [62]. Our approach which makes use of a suitable Gröbner basis of the \( \mathcal{GS} \), though not new, is developed independently. Our approach, given in Algorithm 4.2.2, differs from other approaches based on the Gröbner basis in that it focuses on irrational solution tuples of the \( \mathcal{GS} \) and ignores its rational solutions.

Algorithm 4.2.1 deploys a method for computing solutions of a system of polynomial equations without having to factorize the system every time.

### 4.2.2 Results

Algorithm 4.2.1 computes all equilibria of RPIE games with \( n \geq 3 \) players. We initially show why it does not apply to games with \( n = 2 \) players.

**Proposition 4.** A bimatrix game with all rational payoff values has all rational equilibria.

**Proof.** The \( \mathcal{GS} \) of a bimatrix game is a system of linear equations [57]. Hence, if all the game payoff values are defined over field \( \mathbb{F} \), then all of its solutions can be found in the field \( \mathbb{F} \).

Following is an immediate corollary to the result above.

**Corollary 1.** The class of RPIE games is empty for \( n = 2 \) players.

**Proof.** Follows from Definition 2.3.5 and Proposition 4.

The main result of this section is Proposition 5, which proves the correctness of Algorithm 4.2.1.

**Proposition 5.** Algorithm 4.2.1 for computing all equilibria of RPIE games works. i.e., the output at termination consists of all irrational equilibria of the game, and no other solutions of the \( \mathcal{GS} \).

**Proof.** An input RPIE game \( T \) with \( n \geq 3 \) players is characterized via a \( \mathcal{GS} \) of the form (2.7), which is derived from the inequality on expected payoffs and payoffs at pure strategies. Hence, the \( \mathcal{GS} \) in general has more solutions than just the
In the first phase, Algorithm 4.2.1 calls Algorithm 4.2.2. Algorithm 4.2.2 computes a sample solution $\beta$ by first building a Gröbner basis for the $\mathcal{GS}$ using Buchberger’s algorithm. Buchberger’s algorithm terminates in a triangular form analogous to echelon form in the linear case.

Since the game is known to be RPIE and rational solutions of the $\mathcal{GS}$ are rejected by the Algorithm 4.2.2, the sample solution $\beta$ must have all irrational coordinates. Consequently, each coordinate $\beta_i$ of the sample solution $\beta$ results in an algebraic extension $K = \mathbb{Q}(\beta_i)$ of $\mathbb{Q}$ with finite Galois group $G_i = \text{Gal}(K/\mathbb{Q})$. Since the group action of $G_i$ is transitive, it generates all irrational solutions of the $\mathcal{GS}$.

We know that the $\mathcal{GS}$ has zero-dimensional ideal. This means $\mathcal{GS}$ has finitely many solutions. Group action by a finite Galois group over finite solution set terminates. This enables Algorithm 4.2.1 to reach Step 5 every time there is an RPIE game $T$ as input. The algorithm generates solutions of the $\mathcal{GS}$ that contain all the equilibria of the game $T$.

Finally, Algorithm 4.2.1 rejects solutions of the $\mathcal{GS}$ which are not Nash equilibria. Since the set of Nash equilibria is known to be non-empty, set $X$ contains all and only the Nash equilibrium solutions of the RPIE game $T$.

Proposition 6. If univariate polynomials in the Gröbner basis of an RPIE game have solvable Galois groups, then Algorithm 4.2.1 computes Nash equilibria of the game in closed form.

Note that, Buchberger’s algorithm in first phase of the Algorithm 4.2.2 could be replaced by a numerical method to compute a sample equilibrium.

Recall that in Chapter 1 we mentioned the issue of storing irrational equilibria in computer memory, which has been addressed by Lipton and Markakis [60]. The following result shows that the issue can be resolved for a subclass of RPIE games.

50
4.3. Computational Complexity

Proof. It is a standard result that a polynomial with solvable Galois group is solvable by radicals. If each univariate polynomial in the Gröbner basis of an RPIE game has solvable Galois group, then the roots of this set of polynomials can be computed using radicals. This gives all solutions in closed form, which contains set of Nash equilibria of the game.

It is known that all abelian groups, groups of order $< 60$, groups of odd order (Feit-Thompson Theorem) and groups of order $p^aq^b$, where $p$ and $q$ are prime, are solvable. Moreover, some non-abelian groups are also be solvable. This suggests that Proposition 6 is applicable to a substantial number of games. The equilibria of games with non-solvable Galois groups can be obtained in algebraic form by first computing equilibrium solutions numerically, and then constructing the minimal polynomials of each of the numerical values with the algorithm in [44].

Recall that in Section 3.2.4 of Chapter 3, we mentioned the importance of Proposition 3 for the problem of computing equilibria. In step 9 of Algorithm 4.2.2 we substitute an irrational root $\beta_i$ in the triangular form of the Gröbner basis of the GS. If the substitution produces a solution tuple $\beta$ with mixed coordinates (rational and irrational roots), then we must ignore it. In case Conjecture 1 in Chapter 3 is true for the GS of the input RPIE game, then such rejection and re-computation of solution tuples can be avoided to increase over all efficiency of the Algorithm 4.2.1.

4.3 Computational Complexity

Constructing of the GS (Step 2 of Algorithm 4.2.1) is polynomial time in the size of the input payoff matrix, i.e. polynomial time in $K^*$. A Gröbner basis can be computed in doubly exponential time in the size of $K^+$. A Gröbner basis contains polynomials in triangular form and we are interested in the equilibria points with irrational values. An advantage of the triangular form is

\[\text{It can be verified whether a given polynomial has solvable Galois group or not using the polynomial time Landau-Miller test}^{55}.\]
that at every stage of the substitution unwanted solutions can be filtered out. The
efficiency of Algorithm 4.2.1 could be significantly improved by replacing Buch-
berger’s algorithm with a more efficient method for computing a sample solution.
For further details of computational complexity of finding Gröbner basis, refer
Burgisser and Lotz [9].

We are not considering the issue of computing the Galois groups in this work,
i.e. we consider that the Galois groups are known. But to make the discussion
complete, we give below the complexity of computing the Galois group of a given
polynomial. Computation of a Galois group requires polynomial time in the de-
gree of the input polynomial and the order of its Galois group. If \( f(x) \) is of degree
d then its Galois group can have at most \( d! \) elements and so in worst case the
computation takes exponential time. This is at present best-known upper bound
due to Landau [54]. Lenstra [59] surveys results relating to the complexity of
computing Galois groups and other related problems.

Once a Galois group \( G \) is known, we must find the Galois orbit \( G\beta_i \) of every known
root \( \beta_i \) of every indeterminate variable in the \( \mathcal{G}\mathcal{S} \). An orbit construction takes
polynomial time with the algorithm suggested by Luks [61]. In the worst case,
the algorithm requires action of each of the Galois group generator \( g' \in G' \subseteq G \)
to each element of the set of roots. This gives worst case time \( O(|G'| \cdot |X|) \). If
a univariate polynomial has \( n \) roots, then \( |G'| \) is linear in \( n \) [54, 59], while \( |X| \)
is polynomial in \( n \). Finally, the verification of a Nash equilibrium solution is a
polynomial time operation in the size of total number of strategies \( K^+ \).

The algorithm for computing Nash equilibria via the Gröbner basis, given in [21],
substitutes all the roots in the triangular form and solves univariate polynomial
for each substitution. If the Galois group is known for the polynomials, then
our approach computes solutions with relatively simple and efficient group action.
Our algorithm exploits information available in a sample solution, and performs
better than algorithm which uses only Gröbner basis. If each univariate in the
Gröbner basis of the \( \mathcal{G}\mathcal{S} \) has \( d_i \) distinct roots \( (i \in \{1, \ldots, K^+\}) \), then the method
for computing Nash equilibria in [21] takes \( \Pi_i d_i \) substitutions and factorizations.
4.4. Equilibria Computation of an RPIE Game: An Example

On the other hand, in our approach, after computing a sample solution, no further substitution or factorization is required.

### 4.4 Equilibria Computation of an RPIE Game: An Example

In this section, we show working of Algorithm 4.2.1 with an example of 3 players 2 strategy RPIE game. With the Membership Algorithm 3.1.1 given in Chapter 3, we verify that the game, given in Table 4.1 is an RPIE game.

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>6, -1, 4</td>
</tr>
<tr>
<td>b</td>
<td>0, 3/2, 0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>2, 0, 0</td>
</tr>
<tr>
<td>b</td>
<td>0, 27/2, 0</td>
</tr>
</tbody>
</table>

Table 4.1: Payoff table of a 3-player 2-strategy RPIE game. Player 1 and 2’s strategies are indicated by a, b and A, B respectively. Player 3’s strategies are 1 and 2. Entry in each cell of the payoff table indicates player 1, 2 and 3’s payoff for their respective strategies.

We let $x = x_1^1, y = x_2^1, z = x_3^1$ be the first strategy of players 1, 2 and 3 respectively. The probability that players will choose their second strategy is $1 - x, 1 - y$ and $1 - z$ respectively. The Gröbner Basis for the game is as follows:

\[
2(-1 + x)(-1 + y + z + yz) = 0 \\
2x(-1 + y + z + yz) = 0 \\
-2(-1 + y)(3 + x(-3 + z) - 3z) = 0 \\
-2y(3 + x(-3 + z) - 3z) = 0 \\
\frac{1}{2}(9 - 36y + x(9 + 13y))(-1 + z) = 0 \\
\frac{1}{2}(9 - 36y + x(9 + 13y))z = 0.
\] (4.1)

Next, we apply Buchberger’s algorithm with lexicographical order $x \prec y \prec z$. The Gröbner basis is:

\[
\{-27 + 27x + 5x^2, -18 - 5x + 33y, -15 + 10x + 33z\}.
\] (4.2)
4.5 Discussion

The univariate polynomial $f = -27 + 27x + 5x^2$ has $x = \frac{3}{10}(-9 \pm \sqrt{141})$ as its two roots. Hence, $f$ is irreducible over $\mathbb{Q}$ and has Galois group $\{\text{id,conjugate}\}$ isomorphic to $\mathbb{Z}_2$.

Substituting $x = \frac{3}{10}(-9 - \sqrt{141})$ in the triangular form (4.2) of the Gröbner basis and solving for univariate polynomials in $y$ and $z$ we get: $y = \frac{1}{22}(3 - \sqrt{141}); z = \frac{1}{11}(14 + \sqrt{141})$, a sample solution. The Galois groups of the irreducible polynomials of the $\mathcal{GS}$ are known a priori (isomorphic to $\mathbb{Z}_2$ for each variable $x, y, z$ over $\mathbb{Q}$). All the remaining solutions can be obtained by computing Galois-orbits of the sample solution. The Galois orbits are as follows:

$$
G_x = \{\frac{3}{10}(-9 - \sqrt{141}), \frac{3}{10}(-9 + \sqrt{141})\} \\
G_y = \{\frac{1}{22}(3 - \sqrt{141}), \frac{1}{22}(3 + \sqrt{141})\} \\
G_z = \{\frac{1}{11}(14 + \sqrt{141}), \frac{1}{11}(14 - \sqrt{141})\}. 
$$

(4.3)

In this example, it is sufficient to apply the criteria (2.6) and (2.2) for deciding whether a solution tuple is also an equilibrium solution. Accepting values between 0 and 1, we get $x = \frac{3}{10}(-9 + \sqrt{141}); y = \frac{1}{22}(3 + \sqrt{141}); z = \frac{1}{11}(14 - \sqrt{141}),$ as the unique Nash equilibrium of the RPIE game depicted in Table 4.1. Note that the Galois group for the $\mathcal{GS}$ is solvable, and so, all the equilibria computed are in closed form.

4.5 Discussion

In this chapter we presented an algorithm for computing all equilibria for the class of RPIE games which uses Buchberger’s algorithm and Galois group action. Our computational complexity analysis suggests that most of the time is consumed in computing a sample solution. In the next chapter we address this issue and present a major modification in the method for computing a sample solution. We also discuss the issue: can we generalize the algebra presented in this chapter to consider other classes of games?

---

4For larger systems, the Nash equilibrium verification algorithm [30] comes handy.