Mixed Nash Equilibria Finder for 2-Player Normal-Form Games

Git link:

https://github.com/razvanbaboiucs/BMDC-game-theory-assginment/tree/master/6-mixed-ne

Algorithm Explanation:

Pseudocode:

```
FUNCTION FindCompletelyMixedNE(payoffMatrices)
    // Solve for both players' mixed strategy probabilities
    player1Probability ← SolveIndifferenceEquation(payoffMatrices, player2)
    player2Probability ← SolveIndifferenceEquation(payoffMatrices, player1)

    // Check if valid mixed strategy exists
    IF 0 < player1Probability < 1 AND 0 < player2Probability < 1 THEN
        RETURN "Mixed Nash Equilibrium found" with probabilities
    ELSE
        RETURN "No completely mixed Nash Equilibrium exists"
    END IF
END FUNCTION

FUNCTION SolveIndifferenceEquation(payoffMatrices, player)
    // Extract the 2x2 payoff matrix for this player
    payoffs ← payoffMatrices[player]

    // Set up indifference equation: expected payoff from strategy 1 = expected payoff from
    // For 2x2 games, this simplifies to a linear equation in one variable

    // Calculate mixing probability that makes opponent indifferent
    IF equation has valid solution THEN
        probability ← solution to indifference equation

        // Check if probability is valid (between 0 and 1)
        IF 0 < probability < 1 THEN
            RETURN probability  // Valid mixed strategy
        ELSE
            RETURN invalid result  // Outside valid range
        END IF
    ELSE
        RETURN special case result  // No solution or infinite solutions
    END IF
END FUNCTION
```

```
FUNCTION VerifyEquilibrium(payoffMatrices, mixedStrategies)
    // Extract probabilities from mixed strategies
    p ← probability for player 1's first strategy
    q ← probability for player 2's first strategy

    // Calculate expected payoffs for each pure strategy
    player1Payoffs ← expected payoffs for player 1's strategies
    player2Payoffs ← expected payoffs for player 2's strategies

    // Check indifference condition
    IF player1Payoffs are equal AND player2Payoffs are equal THEN
        RETURN "Verified Nash Equilibrium"
    ELSE
        RETURN "Not a Nash Equilibrium"
    END IF
END FUNCTION
```

Explanation:

1. The algorithm focuses on finding completely mixed Nash Equilibria in 2-player, 2-strategy (2x2) normal-form games.
2. A completely mixed Nash Equilibrium is one where both players use non-degenerate mixed strategies (i.e., they play each of their strategies with non-zero probability).
3. The core principle is based on the indifference condition: in a mixed Nash Equilibrium, each player must be indifferent between all pure strategies in their support.
4. For a 2x2 game, this means:
    - Player 1's expected payoff from playing strategy 1 equals their expected payoff from playing strategy 2
    - Player 2's expected payoff from playing strategy 1 equals their expected payoff from playing strategy 2
5. The algorithm solves these indifference equations to find the mixing probabilities for each player.
6. It handles various special cases, such as when no solution exists, when solutions are outside the valid probability range, or when the game has infinitely many mixed equilibria.
7. A verification step confirms that the found mixed strategy profile is indeed a Nash Equilibrium by checking that each player is indifferent between their pure strategies.

Mathematical Foundation:

- For a 2x2 game with payoff matrices A (for Player 1) and B (for Player 2), let:

  - p = (p , p) be Player 1's mixed strategy, where p + p = 1
  - q = (q , q) be Player 2's mixed strategy, where q + q = 1
- The indifference conditions are:
  - For Player 1: q · A + q · A = q · A + q · A
  - For Player 2: p · B + p · B = p · B + p · B
- Solving these equations and simplifying:
  - q = (A - A) / (A - A - A + A)
  - p = (B - B) / (B - B - B + B)
- For a completely mixed Nash Equilibrium, both p and q must be in the open interval (0, 1).

Data Structure:

- The algorithm uses 2D arrays to represent payoff matrices for the 2-player game
- Each payoff matrix is a 2x2 grid where:
  - payoffMatrices[0][i][j] represents Player 1's payoff when Player 1 plays strategy i+1 and Player 2 plays strategy j+1
  - payoffMatrices[1][i][j] represents Player 2's payoff when Player 1 plays strategy i+1 and Player 2 plays strategy j+1
- The mixed strategies are represented as arrays of probabilities, where:
  - strategies[0] = [p , p] represents Player 1's mixed strategy
  - strategies[1] = [q , q] represents Player 2's mixed strategy

Time and Space Complexity:

- Time Complexity: O(1) for finding mixed Nash Equilibria in 2x2 games
  - The algorithm solves a fixed-size system of linear equations
  - Verification also takes constant time
- Space Complexity: O(1) for storing the payoff matrices and mixed strategies
  - The space required is independent of the input size since we're dealing with 2x2 games

Example Games

Game 1: Matching Pennies

Mixed Nash Equilibria Finder for 2-Player Normal-Form Games

This implementation finds Nash Equilibria for 2x2 games. Do you want random payoffs? (yes/no): no

Enter payoffs for Player 1: Enter payoff for strategy profile (1, 1): 1 Enter payoff for strategy profile (1, 2): -1 Enter payoff for strategy profile (2, 1): -1 Enter payoff for strategy profile (2, 2): 1

Enter payoffs for Player 2: Enter payoff for strategy profile (1, 1): -1 Enter payoff for strategy profile (1, 2): 1 Enter payoff for strategy profile (2, 1): 1 Enter payoff for strategy profile (2, 2): -1

Payoff Matrix for Player 1: [[1, -1] [-1, 1]]

Payoff Matrix for Player 2: [[-1, 1] [1, -1]]

Completely Mixed Nash Equilibria: Found a completely mixed Nash Equilibrium: Player 1's strategy: (0.5000, 0.5000) Player 2's strategy: (0.5000, 0.5000)

Verification: Is this a Nash Equilibrium? Yes

Expected Payoffs: Player 1 - Strategy 1: 0.0000 Player 1 - Strategy 2: 0.0000 Player 2 - Strategy 1: 0.0000 Player 2 - Strategy 2: 0.0000

Game 2: Battle of the Sexes

Mixed Nash Equilibria Finder for 2-Player Normal-Form Games

This implementation finds Nash Equilibria for 2x2 games. Do you want random payoffs? (yes/no): no

Enter payoffs for Player 1: Enter payoff for strategy profile (1, 1): 3 Enter payoff for strategy profile (1, 2): 0 Enter payoff for strategy profile (2, 1): 0 Enter payoff for strategy profile (2, 2): 2

Enter payoffs for Player 2: Enter payoff for strategy profile (1, 1): 2 Enter payoff for strategy profile (1, 2): 0 Enter payoff for strategy profile (2, 1): 0 Enter payoff for strategy profile (2, 2): 3

Payoff Matrix for Player 1: [[3, 0] [0, 2]]

Payoff Matrix for Player 2: [[2, 0] [0, 3]]

Completely Mixed Nash Equilibria: Found a completely mixed Nash Equilibrium: Player 1's strategy: (0.6000, 0.4000) Player 2's strategy: (0.4000, 0.6000)

Verification: Is this a Nash Equilibrium? Yes

Expected Payoffs: Player 1 - Strategy 1: 1.2000 Player 1 - Strategy 2: 1.2000 Player 2 - Strategy 1: 1.2000 Player 2 - Strategy 2: 1.2000

Game 3: Prisoner's Dilemma

Mixed Nash Equilibria Finder for 2-Player Normal-Form Games

This implementation finds Nash Equilibria for 2x2 games. Do you want random payoffs? (yes/no): no

Enter payoffs for Player 1: Enter payoff for strategy profile (1, 1): 3 Enter payoff for strategy profile (1, 2): 0 Enter payoff for strategy profile (2, 1): 5 Enter payoff for strategy profile (2, 2): 1

Enter payoffs for Player 2: Enter payoff for strategy profile (1, 1): 3 Enter payoff for strategy profile (1, 2): 5 Enter payoff for strategy profile (2, 1): 0 Enter payoff for strategy profile (2, 2): 1

Payoff Matrix for Player 1: [[3, 0] [5, 1]]

Payoff Matrix for Player 2: [[3, 5] [0, 1]]

Completely Mixed Nash Equilibria: No completely mixed Nash Equilibrium exists for this game. Completely Mixed Nash Equilibria: The calculated probabilities are outside the valid range [0,1].

Game 4: Random Game

Mixed Nash Equilibria Finder for 2-Player Normal-Form Games

This implementation finds Nash Equilibria for 2x2 games. Do you want random payoffs? (yes/no): yes

Payoff Matrix for Player 1: [[63, 63] [84, 46]]

Payoff Matrix for Player 2: [[50, 51] [57, 0]]

Completely Mixed Nash Equilibria: Found a completely mixed Nash Equilibrium: Player 1's strategy: (0.9828, 0.0172) Player 2's strategy: (0.4474, 0.5526)

Verification: Is this a Nash Equilibrium? Yes

Expected Payoffs: Player 1 - Strategy 1: 63.0000 Player 1 - Strategy 2: 63.0000 Player 2 - Strategy 1: 50.1207 Player 2 - Strategy 2: 50.1207

Limitations

Current Limitations:

1. The algorithm is specifically designed for 2-player, 2-strategy (2x2) games
2. It only finds completely mixed Nash Equilibria, not pure or partially mixed equilibria
3. Numerical precision issues may affect the accuracy of solutions in some cases

Generalization to 2-player, n-strategy (nx2) games

Generalizing the algorithm for finding completely mixed Nash Equilibria to games with any number of strategies for both players (beyond 2x2) involves a significant increase in complexity, primarily because the indifference conditions become a system of linear equations rather than simple 2x2 equations.

Here's a conceptual explanation of how it would work:

1. *Indifference Conditions for N Strategies*: For each player, if they are playing a mixed strategy, they must be indifferent between all pure strategies they play with non-zero probability. If a completely mixed Nash Equilibrium exists, then each player must be indifferent between all of their pure strategies. This means for a player with m strategies, you'd have m-1 indifference equations.

2. *System of Linear Equations*: For a game with two players, Player 1 with m strategies and Player 2 with n strategies:

   - Player 1's indifference conditions will generate m-1 equations involving Player 2's n probabilities.
   - Player 2's indifference conditions will generate n-1 equations involving Player 1's m probabilities.
   - Additionally, the probabilities for each player must sum to 1 (e.g., p1 + p2 + ... + pm = 1 and q1 + q2 + ... + qn = 1).

3. *Solving the System*: The core challenge is solving this larger system of linear equations. Unlike the 2x2 case where simple algebraic manipulation suffices, for m x n games, you would typically need to use more advanced linear algebra techniques