An Algorithmic Approach to Grid Game

Aryan Prodduturi
Georgia Institute of Technology
Atlanta, USA
aryanpro@gatech.edu

Ananya Jain
Georgia Institute of Technology
Atlanta, USA
ananyaj@gatech.edu

Saloni Bedi
Georgia Institute of Technology
Atlanta, USA
sbedi9@gatech.edu

***Abstract*—This paper describes a two-player game that uses a grid and pieces consisting of a number and a color in which each player attempts to eliminate possible moves for the other player at each turn. The paper then discusses a set of rules that, if followed by the first player, can assure that they will win the game on a 3×3 board.**

I. Background

In this project, a two-player game is developed in which the goal of one player is to make it such that the other player has no possible moves on an $n \times n$ grid. Every game, the players start with a total of $n \times n$ "pieces" which are combinations of a color and a number from n colors and n numbers. The player places these (number, color) pieces on a cell on the game board with the following rules in mind:

- Adjacent pieces must match in color or number.
- After a player has used a specific (number, color) pair, it cannot be used again by either player.

Throughout the game, the goal of both players is to minimize the possible cells the other player can place the remaining pieces on until they cannot place any of the remaining pieces on any cell.

II. Upper Bound on the Time Complexity

The complexity of the game can be estimated using an upper bound for the number of different positions that can be achieved. On a $n \times n$ board, there are a total of $(n \times n + 1)!$ possible position. With an empty game board, the first player can choose to place any of the $n \times n$ pieces in a specific cell or leave it empty. Once this player uses a piece and plays, the second player can place any of the remaining $(n \times n - 1)$ pieces in a specific cell or leave it empty. This pattern continues until there are two possibilities for the last cell: empty or the last piece is used. The product of these possible positions is $(n \times n + 1)!$, the 1 accounting for the possibility of a cell being empty.

This upper bound for a 3×3 board using this formula is 3,628,800, which can be lowered if illegal positions are considered. For example, a player cannot place R1 next to G3 because these pieces match in neither color nor number. In this way, if a player places a piece on an edge, they limit the number of possible positions of the adjacent cells to 2n-1 and 2n-2. Using the previous example, placing an R1 in the upper left corner limits the possibilities for the right cell to R2, R3, G1, B1, or empty, and for the left cell to this number minus one if the right cell is occupied first. If this logic is applied to one of the center cells, the possible positions for the adjacent cells become 2n-1, 2n-2, 2n-3, and 2n-4 assuming they are used one after the other and these are the first moves of the game. However, the smaller value for this bound is difficult to estimate as the number of illegal positions and unavailable squares (those that cannot be played on based on the adjacent pieces) change based on how the players choose their pieces and/or the cells they place these pieces on.

A. Example

Assume the game board is 3×3 in size and the pieces are formed from the colors: red, blue, green and numbers: 1, 2, 3. The cells are numbered left to right, top to bottom.

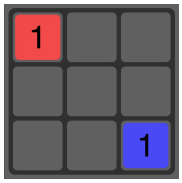

Fig. 1. Iteration 1: Player 1 places R1 in cell 1 and player 2 places B1 in cell 9.

Fig. 2. Iteration 2: Player 1 places G1 in the middle of the board (cell 5). Now Player 2 cannot place anything in cells 2, 4, 6, or 8 because the position of these cells relative to the pieces on the board require a piece with a 1, but these have all been used up (R1, B1, and G1). So Player 2 places R3 in cell 3.

Fig. 3. Iteration 3: Player 1 places B2 in cell 7. Player 2 cannot place any of the remaining pieces on the board as cells 2, 4, 6, and 8 cannot be used according to the aforementioned reason. Therefore, player 1 has won!

III. Main Result

The paper aims to find the perfect play, i.e. under the given restriction – number of colors, number of digits, and the size of the grid – which player is guaranteed to win. Starting off with a 3×3 grid with the choices belonging to $N \times C$, where N = 1,2,3 and C = R,B,Y. In this situation it is true that the first player always has a way to win if he plays the correct moves.

This can be analyzed by considering the number of available spots to play once a player makes his move. An available spot is a square that can contain one of the remaining pieces, and an unavailable spot is a square that has no playable piece. For example, a position adjacent to R1, B2, and Y3 is unavailable because no single piece can meet the adjacency requirements for all of these pieces. There need to always be an even number of available spots after the first person plays in order for them to win. And it can be seen that there will always be a move where there are an even number of boxes remaining, as from the start the control of blocking spots is in player 1's hand, and he can ensure even playable spots after his turn.

To ensure there is an even number of available spots after the first player's turn, they will need to block off a number of spots that is the same parity as the second player blocked off on the previous turn. This works because two numbers of the same parity with always add to an even number, after each time both players finish their turns the parity of available spots does not change. This strategy can be followed for all but the first turn, since the second player has not played a move yet, in which case the first player can play anywhere resulting in 8 available spots.

Fig. 4. After Player 1 plays R1 in cell 1 and Player 2 plays B1 in cell 9, Player 1 places G1 in cell 6 and blocks off 4 adjacent cells. There are now an even number of spots left which can guarantee a win for Player 1.

A. The Algorithm

To guarantee a win the first player can follow the following strategy. First they can play any piece into the middle square, it doesn't matter what piece is chosen so assume they play the B2 piece. From now on they must play pieces in the location across from where the second player plays. For example, if the second player plays in the top left, the first player should play in the bottom right. The specific piece they play must be blue if the previous play was blue or the remaining color if red or yellow was chosen. Additionally that piece must be the 2 if the previous play was 2, or the remaining number if 1 or 3 is chosen. This system pairs up all the remaining pieces like so:

- B1 $\leftrightarrow$ B3
- R1 $\leftrightarrow$ Y3
- R2 $\leftrightarrow$ Y2
- R3 $\leftrightarrow$ Y1

Since all the remaining pieces are paired 1-to-1, if one of the pieces in a pair is remaining the other piece must be remaining too. Because of this symmetrical placement and piece selection, the first player must always have a move in response to the second player and therefore cannot lose the game.

IV. Extension

The example game focuses on the case of a 3×3 board with 3 colors and 3 numbers, where n=3. This game can be extended to an $n \times n$ grid with n colors and n numbers. The algorithm previously explained can also be extended to a grid of size n.

The possible pieces are $C \times N$ where C is the set of colors and N is the set of numbers. The elements of C are defined as $C_1, C_2 \ldots n$ and the elements of N are defined as $1, 2 \ldots n$ Both C and N have a cardinality of n resulting in n^2 pieces. I will now provide algorithm that will guarantee a win regardless of the value of n.

When n is odd, the first player should use this algorithm to win. Since n is odd, n^2 is also odd. After the first move, one square on the board will be filled resulting in an even number of resulting squares. The first move played should be $\{C_p, p\}$ in position (p, p), where $p = \lceil n/2 \rceil$. Notice that $\lceil n/2 \rceil$ is the "middle" choice in all categories.

When n is even, the second player should use this algorithm to win, and therefore it is not possible to play the first move.

From this point, the move to be played is only based on the opponents previous move. If the opponent plays the piece $\{C_a, b\}$ where in $0 < a \leq n$ and $b \in N$ at the position (x, y) where $0 < x \leq n$ and $0 < y \leq n$, then the move to be played in response is $\{C_{n+1-a}, n+1-b\}$ at position $(n+1-x, n+1-y)$. Notice that all 4 parts of the move are replaced with their respective $(n+1)$ complement.

Since $(n+1)$ complements come in pairs, every piece also has a complement with the piece that has both values which are $(n+1)$ complements of the original one. Using the same logic, every position on the board has a complement where both the x and the y are $(n+1)$ complement of the original position. This guarantees that for every move the opponent plays, the algorithm user has another move to play. Since the only way to lose in this game is to not have a playable move, this guarantees that the algorithm user wins.

A. Strategies

While it's possible to guarantee wins using the extended algorithm, there are still other strategies to be used when someone is playing as the opponent. As explained in the Main Result, you can used the "available moves" to know which moves are better to play. Playing a move that changes the parity of the number of available moves is not useful if the opponent can do the same. Therefore you must look ahead moves to see if the parity is actually changed long term. The

higher depth you look ahead, the better however that also comes at the cost of computation time.

There are also more advanced ways to create blocked spaces that a more complex strategy can consider like playing the only piece that could go a specific location somewhere else, or restricting multiple spots to a smaller set of pieces.

For example, if there's a spot that can only hold R3, you can play the R3 in a different location to increase the number of unavailable spots. This spot is denoted in the following image with the star.

On this board, if the first player then played R3 in the bottom right square, that would increase the number of unavailable spots even though the move didn't restrict and squares around it.

Another example is if two spots can only hold B1, so one or both of those must become a unavailable spot at some point in the game. These spots are denoted in the following image with the stars.

When counting the number of unavailable spots on this board, we should count both of these squares together as one unavailable spot, showing that the first player would win because there's a even number of them. If we did not make this adjustment, there would have been 2 available falsely indicating that the second player would win.

V. Code

The code is currently hosted on github at https://github.com/1aryanpro/grid-game and the game can be played at https://1aryanpro.github.io/grid-game/.

References

[1] V. Allis, "A Knowledge-based Approach of Connect-Four" Department of Mathematics and Computer Science. Vrije Universiteit.