1. **Introduction**
   1. **Race games**

Race game is a category of board games in which the aim is to move the designated pieces assigned to a particular player to the end of a track. These games are often played using a die and requires strategic thinking on how to win the game.

* 1. **Ludo**

Ludo is a complex race game which combines luck and skill as compared to simple ones such as snake and ladder, which is purely based on luck. It is a non-deterministic game with about 2-4 players, where each player is assigned a color from blue, red, green and yellow, and has 4 pieces and a path to move along from their “start” position to their “home” position. The movement of the pieces is determined by the roll of a die and the player's choice of which piece to move. A player wins when all of their pieces are at their respective home base.

* 1. **Reinforcement learning Algorithm**

It is also known as the science of decision making. It is a part of machine learning which uses optimal behavior, learned by interacting with the environment and recording its response. The purpose of this algorithm is to take actions based on its observations and learn through trial and error.

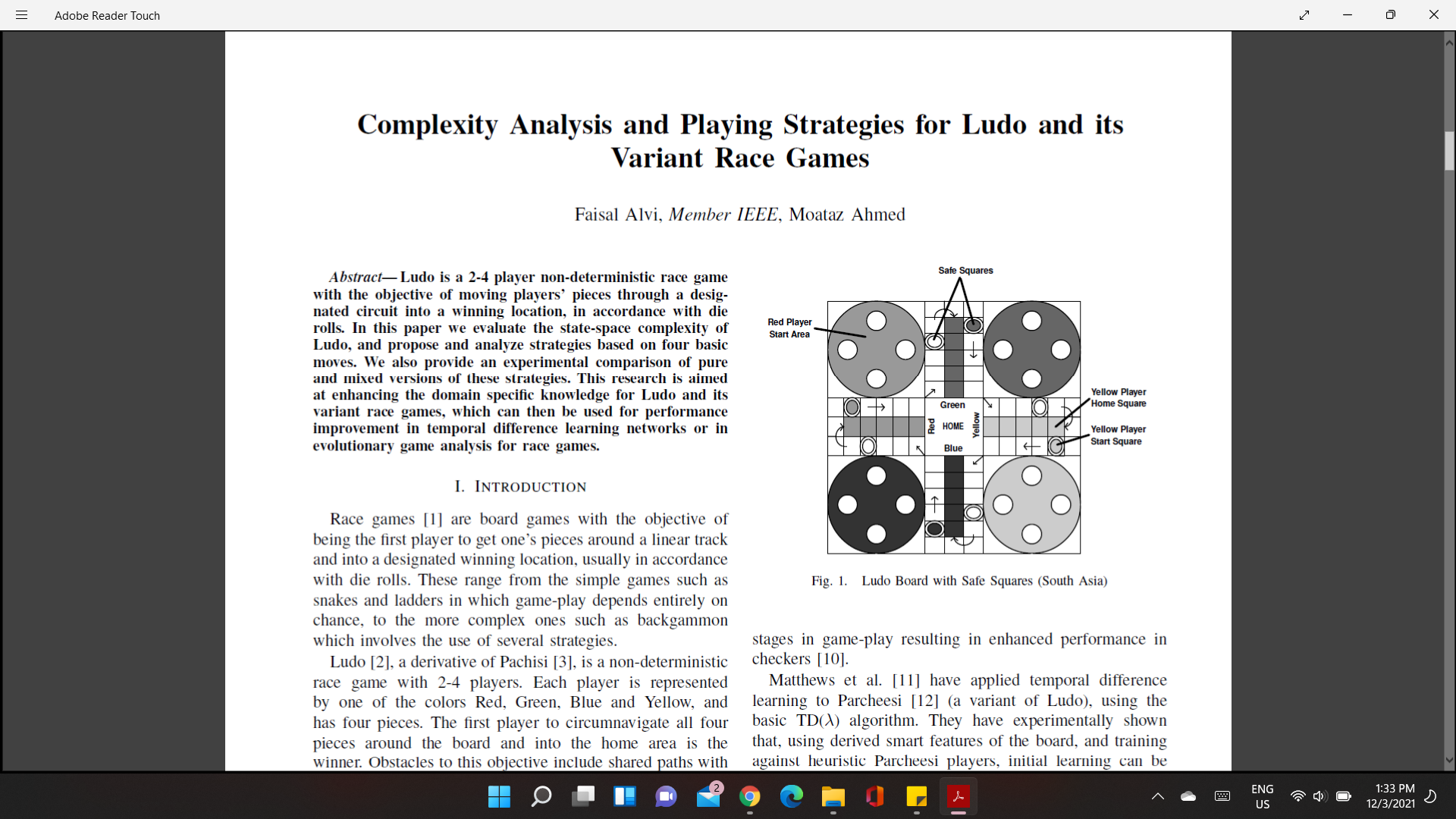
* + 1. **Temporal difference (TD) learning Algorithm**

It is an algorithm that falls under reinforcement learning. TD learning takes place without a predefined model. It is similar to learning from trial and error since it learns directly from raw experience without any prior knowledge about the model. It estimates results simultaneously as it is learning rather than waiting for the outcome. Therefore, it is a favorable methodology for developing game-playing agents in board games.

1. **State space complexity in ludo**

State space complexity is defined as the number of legal game states reachable from the initial state of the game. The primary application of state-space complexity is to determine whether a perfect evaluation function can be constructed using table look-up by listing and evaluating every possible state of a game. To date no attempt has been made to compute the complexity of a generalized version of Ludo; several variants of the game may give some hints in that direction. In the absence of such a generalization, it is possible to rule out Ludo being easily solved by listing all possible game states

* 1. **Assumptions**

By considering that all 16 pieces (4 players \* 4 pieces/player) can be placed anywhere on the board, we can establish a trivial upper bound of . This comes out to 5716 ~ 1028 for the board. This figure does not take the game rules into consideration. Below are some assumptions based on rules and observations of the game that need to be taken into account while calculating state space complexity.

* + 1. We categorize the common positions on a Ludo board into two types: safe squares and non-safe squares. The concept of a safe square refers to the locations on the game board where pieces from different players can be placed concurrently without being knocked off. A non-safe square, however, can only accommodate one player at a time.
    2. All pieces of a player are considered to be identical as there is no hierarchy between individual pieces of the same player. Hence a board configuration in which red’s pieces are on locations (3, 4, 2, 1) is identical to another configuration in which red’s pieces are on locations (4, 3, 2, 1).
    3. A safe square can be considered an imaginary single square made up of the four start position squares for each player, along with five home squares each for each player; likewise, the final home location can also be considered a safe square. As a result, the number of computed safe squares is defined as the sum of the actual number of safe squares with these mapped safe squares.
  1. **Theorem**

**For Upper Bound:**

Theorem 1: For a Ludo board, let

*ns* = number of actual safe squares,

*no* = number of non-safe squares,

*nh* = number of home squares for any player,

*nc* = number of computed safe squares = *ns* + *nh* + 2,

r, g, y, b = number of pieces of Red, Green, Yellow, Blue, player on non-safe squares at any given instant,

*f*(*no, nc*) =

**Proof**: Let r Red Player pieces be on any of the no non-safe squares of the board at any given time, and the other 4-r pieces be on any of the calculated safe squares. These r pieces can be placed in C(, r) ways on any of the board's no squares, while the remaining 4-r pieces can be placed on the calculated safe squares.

Red’s pieces can be placed in .

The number of possible locations for Green is given by .

For Yellow, the locations are given ways and similarly, for Blue, in ways.

**Corollary**: The state-space complexity of a Ludo-game has a lower bound of *f*(*no* − 4, *nc*)

The computation of the expression for the upper bound includes one illegal non-safe position for each component, as indicated in the introduction to Theorem 1. We get the phrase for lower bound by omitting all such sites (4 in total).

We find that the state-space complexity of Ludo using the board as shown is 1022. This can be obtained by substituting the values, *no* = 44, nh = 5, *ns* = 8, ⇒ *nc* = 5 + 8 + 2 = 15 in equation (1), and getting a resulting value of ⇒ f(*no*, *nc*) = 1.6 × 1022 . Similarly a lower bound of *f*(*no* − 4, *nc*) = 5 × 1021 is also computed. It is clear that the lower bound differs from the upper bound by a factor of 10 only.

1. **Strategy**

Ludo's game tree branching factor is 24, which equates to four pieces and six alternative die rolls for each player. A player has four possibilities for moving his pieces after he has rolled a die in his move. The following are the many types of moves a player can make throughout a game. Following that, we give a variety of ways based on these types of maneuvers.

* 1. **Random**

A player decides to play one of his pieces completely at random in a random move. When a player sees no benefit in moving a specific piece, he or she may make this move.

A random strategy is one in which the player makes random moves throughout the game. Although a random strategy is ineffective at winning games, it can be used to compare different approaches.

* 1. **Aggressive**

In an aggressive move, a player likes to move a piece that, depending on the die roll outcome, can knock out or eliminate the piece of another player. An aggressive move imposes an additional overhead on the attacking player, since he or she must now re-play his or her eliminated piece across the full game board. As a result, an aggressive move gives the aggressor a clear advantage over the victim.

When possible, an aggressive strategy is centered on aggressive moves throughout the game. When there is no likelihood of assaulting another player's piece, a player may use a different form of move (for example, a random move).

* 1. **Defensive**

A player always defends his pieces in a defensive move against the threat of an incoming onslaught. Though the danger of being attacked is not certain, we can propose that a piece is in danger of being deleted if it is between 1-6 squares of another piece and is smaller than the distance of a single die roll. This distance is known as the knocking range. As a result, if a player's piece is within knocking range of another player's piece, he or she makes a defensive move.

A good defensive move offers the defender an advantage over all other players, as the defending player has averted a loss versus all other players by protecting his piece. When possible, a defensive strategy is predicated on a preference for defensive moves.

* 1. **Fast**

In a fast move, a player selects the piece that has traveled the greatest distance in its circuit around the board. A quick move is based on the premise that losing a piece that has advanced the most in the game (i.e., the fastest piece) would be the costliest to a player (in terms of further movements required), hence it should be moved first and sent to its final home location.

A fast move reorders a player's pieces' movement inside the game and so provides no comparable advantage over other players. However, because a player's just one piece is active (moving) at any given time, the risk of piece elimination is lessened. In this sense, it can be described as a depth-first game, because the player must always consider the depth of the situation.

The player would aim to make his most advanced piece in the game reach the home base first in this approach. Hence, he/she must make quick decisions.

* 1. **Mixed**

At different phases of the game, a player might choose to play different sorts of moves, which is possible and favorable. A player may, for example, use any mixture of defensive, aggressive, rapid, and random actions, resulting in a mixed or hybrid strategy.

We want to underline that the sorts of actions and strategies provided here are by no means comprehensive, and that other methods or types of moves may be discovered at different phases of the game.

1. **Analysis of the strategies**

Over a course of multiple games of ludo, an analysis was done to figure out the advantage of each proposed strategy over the other. The factors taken into account during this analysis are- Expected values of die rolls and average piece movement through the course of the games. For a clearer understanding of the analysis, the following terms have been defined.

1. **Move**

A Move refers to the movement of one of the player's pieces across the designated track. The number of spaces a particular piece can move is determined by the roll of the die, with a maximum of 6 and a minimum of 1. There is an additional "move" when a player rolls a 6, as they get another chance to roll the die.

1. **Distance**

Distance refers to the position of the piece from the home base. it is determined by the number of squares a piece has covered. the total distance a piece has to cover is denoted by board .

1. **Winner**

A winner in ludo is the first person to send all 4 of their pieces to their final home. The winner does this by using the least number of moves.

* 1. **Expected number of moves**

Despite ludo being a non-deterministic game where there is no limitation on the number of moves. A model can be created to record the game progress based on the expected number of moves over multiple games.

Taking an ideal environment into consideration where a piece is not eliminated by another, the minimum number of expected moves taken to move a distance d by a piece is

where 3.5 is the equal chance of getting each number from 1 to 6.

Therefore, for one piece to cover a distance of 57 spaces, =16.3 moves, so a player will need a minimum of 260 moves.

Considering an actual game in ludo, a player’s piece may get knocked off multiple times. Assuming that a player p’s piece *i* gets knocked off k times, the minimum expected number of moves is given by

where dik is the distance covered by the piece. Hence a player may require

minimum moves to win the game

* 1. **Comparison of strategies**

From the 1st equation it is observed that is a constant. Hence there are 2 ways for an expected winner to minimize his/her moves (Ep[m]):

1. Follow a defensive strategy to reduce the number of their own pieces,
2. Follow an aggressive strategy and increase other players’, , by knocking their pieces off.

When compared, it can be concluded that a defensive strategy is most effective as it gives the player an advantage over the other players by not increasing the expected number of moves and keeping their pieces comparatively safe. An aggressive strategy on the other hand is only effective in incrementing the number of moves of one player, leaving 2 more competitors against. A fast strategy is effective in terms of knocking out an advance piece which is close to victory. Whereas a random strategy has no particular advantage, however it can be used to benchmark the other strategies.

**DEFENSIVE ≽ AGGRESSIVE ≽ RANDOM≽ FAST ≽ RANDOM**

1. **Conclusion and future work**

Through this research we reached a conclusion that ludo has a state space complexity of 1022 moves, based on the analysis done. However, in order to fully solve the game, more processing resources are required. In order to limit the number of moves, four tactics were developed: defense, attack, quick, and random. As a result, it was deduced that defense is the most effective. In addition, a mixed strategy was developed that included the fundamental techniques and was found to be more effective than the basic strategies. Furthermore, the state space complexity and strategy analyses can be applied to a wide range of ludo versions.

The results of the various methods' analyses can be fed into the TD learning algorithm to get better results with fewer trials. It could be useful in game analysis for improving and discovering strategies and game play. This is accomplished by assigning weights to moves made in accordance with the strategy and its hierarchy of advantage.

Evolutionary algorithms, which are based on the positioning and pieces in play in ludo, could be used for improvement of the evaluation functions after receiving successive generations. This could result in the discovery of new strategies in the future for improving game play. Moreover, reinforcement learning, the evaluation functions can be adjusted based on the results received from a few initial game runs.

1. **Experimentation and results**

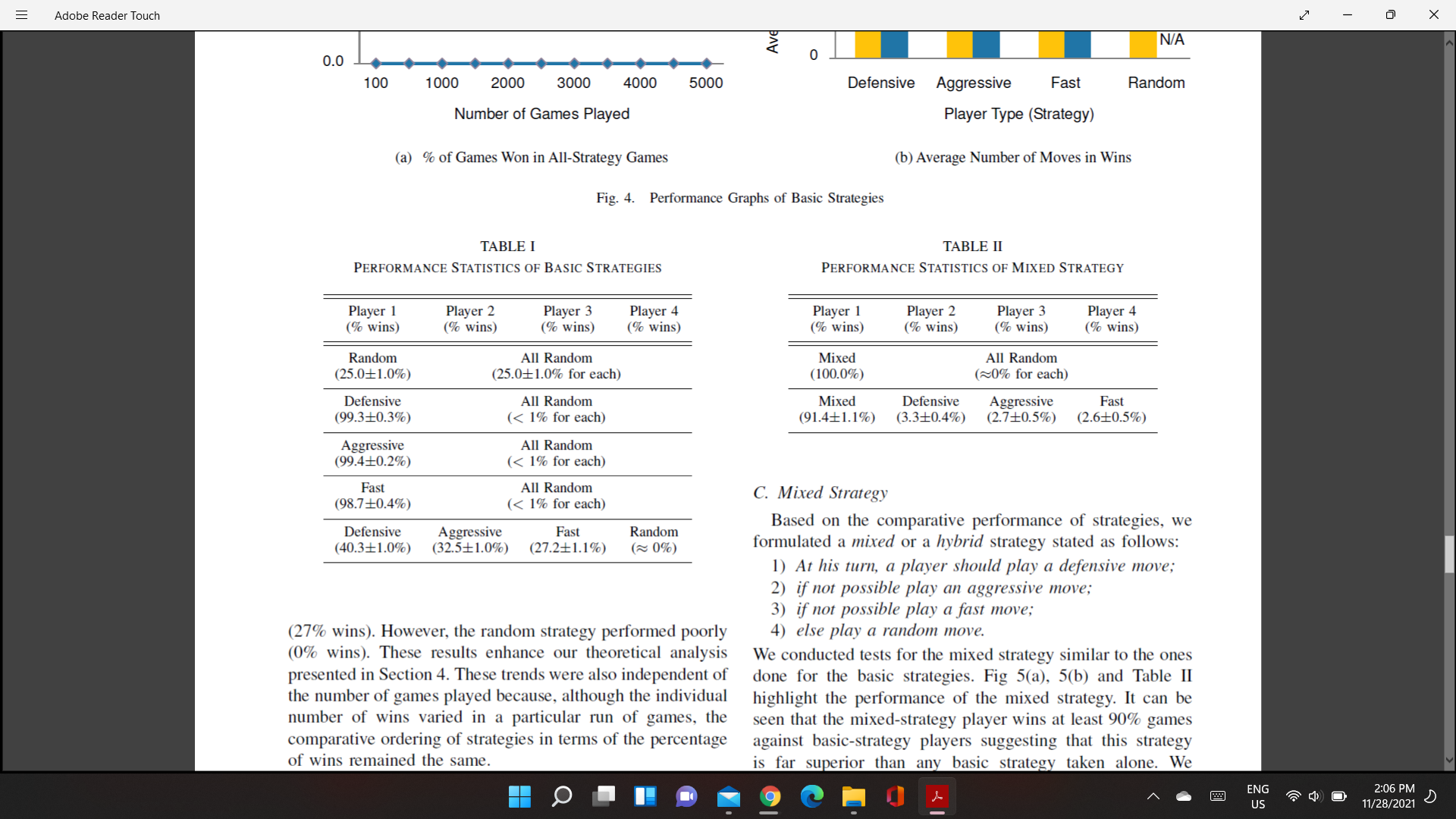
The experiments on the recommended strategies, as well as the findings achieved, are described in this part. These tests were performed on (a) every basic-strategy player playing against all random players, and (b) every basic-strategy player playing against various basic-strategy players. Based on the findings, a mixed approach was developed and evaluated against these fundamental tactics.

* 1. **Setup**

The game was created by creating classes that represented the various entities seen in a standard Ludo game. We first put our gaming setup to the test by playing multiple games with all four players using the random strategy. This was done to confirm that our game setup was correct and that each random player won around the same number of times. We discovered that each random participant wins 25.0∓1.0 percent of the time, and that this performance leveled off at around 5000 games.

The number of games played did not decrease as the number of games increased. There was a considerable difference in the outcomes, implying that there were 5000 games in all. There was a sufficient number for detecting consistent trends in strategy appraisal. When the player order was changed, the findings remained the same, implying that there was no significant bias in favor of the person who took the first turn. The experiments were also conducted separately for each technique, yielding the same results.

* 1. **Evaluation of Strategy Performance**

We next put each of the offered strategies through its paces in the next phase. Two types of testing were used in this phase: We conducted tests on each of the candidates in the next round. There were two types of tests during this time:

* Individually testing each basic-strategy player against three random players to observe the strategy's individual performance data.
* Playing all four basic-strategy players against each other in a game to see how each approach performs relative to the others.

For the first kind, we pitted each basic-strategy player against all other players at random. We discovered that against all random players, each of the basic-strategy players (defensive, aggressive, or rapid) wins at least 98 percent of the games.

Table I shows the outcomes of these testing. This test clearly shows that using any fundamental strategy is preferable to playing at random.

Individually testing each basic-strategy player against three random players to observe the strategy's individual performance data.

• Playing all four basic-strategy players against each other in a game to see how each approach performs relative to the others.

For the first kind, we pitted each basic-strategy player against all other players at random. We discovered that against all random players, each of the basic-strategy players (defensive, aggressive, or rapid) wins at least 98 percent of the games.

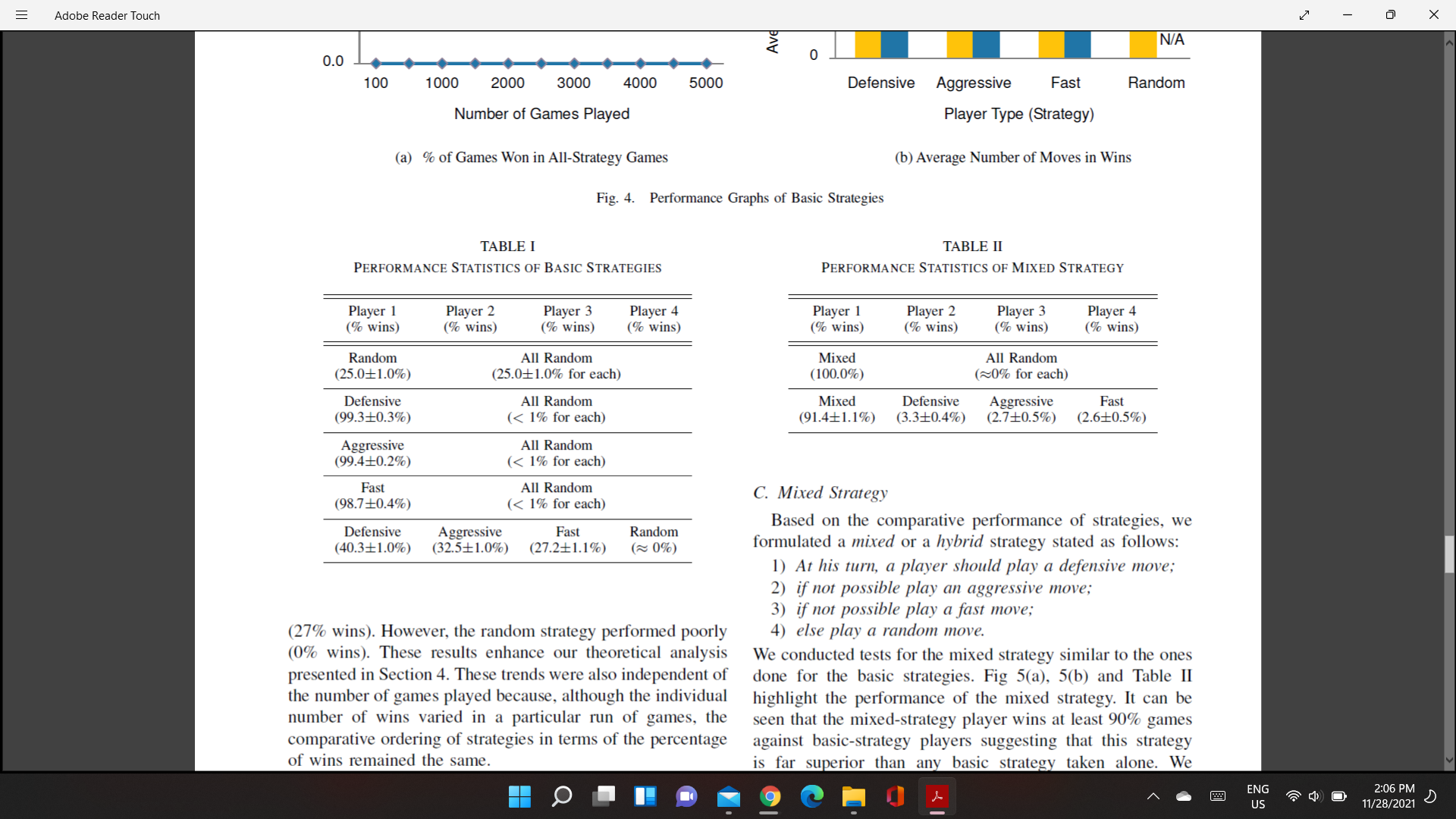
Table I shows the outcomes of these testing. This test clearly shows that using any fundamental strategy is preferable to playing at random.

We pitted all basic-strategy players against each other in multiple game runs for the second sort of test. Figure 4(a) depicts the test findings graphically. Figure 4(a) shows that the defensive strategy consistently beat other strategies (40 percent wins), while the aggressive strategy (32 percent wins) excelled the quick strategy.

(Winning percentage: 27%) The random strategy, on the other hand, failed miserably (0 percent wins). These findings add to our theoretical analysis in Section 4. These tendencies were also unaffected by the number of games played because, while the individual number of victories in a given run of games fluctuated, the relative order of tactics in terms of percentage of wins remained constant.

Figure 4(b) shows the average number of moves taken by each basic-strategy player to win games. Despite winning a bigger number of times, the defensive player took a comparably smaller number of moves to win against other-strategy players. The aggressive and quick strategies required about the same number of moves. It's also worth noting that each participant made more movements versus all-random players than against players with strategy, which appears counterintuitive at first.

This is hardly surprising, given that players won over 98 percent of the time against all-random opponents, which may have included longer games.

* 1. **Mixed Strategy**

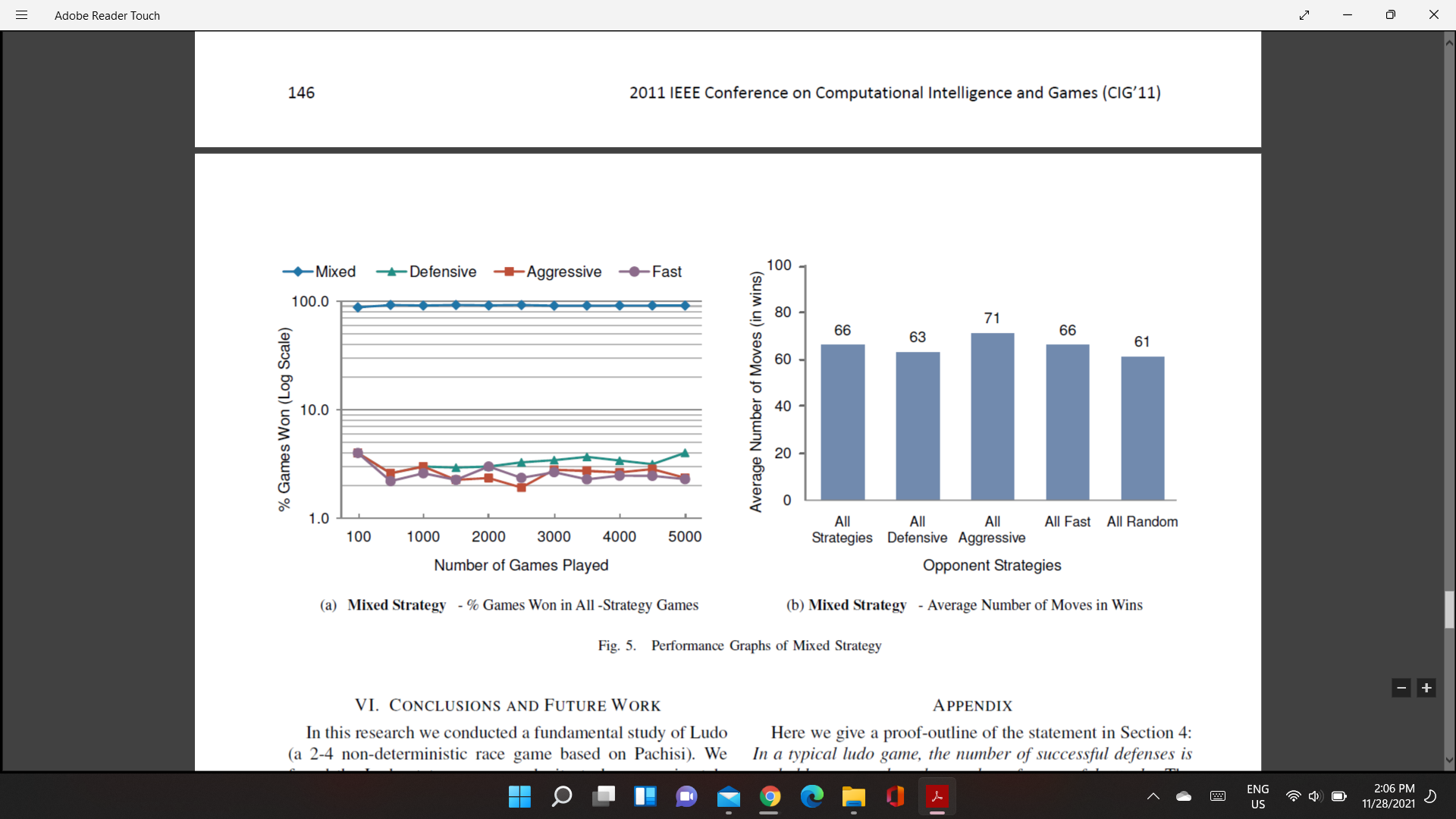
We compare the performance of methods based on their comparative performance and formulated the following mixed or hybrid strategy:

1) A player should make a defensive move during his turn;

2) If at all feasible, make a bold move;

3) If at all possible, make a quick motion;

4) If everything else fails, make a random move.



We tested the mixed approach in the same way that we tested the fundamental methods. The mixed strategy's performance is highlighted in Figures 5(a), 5(b), and Table II. As can be seen, the mixed-strategy player wins at least 90% of games against basic-strategy players, implying that this technique is considerably superior to any single basic strategy.

The average number of moves taken by the mixed-strategy player against other basic-strategy players is close to 65, which is the estimated minimum number of moves required to win (as detailed in Section 4). As a result, the mixed strategy is close to being an optimal approach versus basic-strategy players, as it takes an estimated minimal amount of moves to win a game on average.

The preceding experimental analysis applies to Parcheesi and other Ludo variants because the only difference between them is the number of squares on the game board and a few game regulations; nonetheless, the core game structure remains the same.

1. **Glossary**

**L:**

Lower Bound: An element less than or equal to all the elements in a given set.

**R:**

Reinforcement learning Algorithm: It is referred to as the science of decision making.

**S:**  
State space complexity: State space complexity is defined as the number of legal game states reachable from the initial state of the game.

Strategy: Method to achieve a particular goal.

**T:**

Temporal difference (TD) learning Algorithm: It estimates results simultaneously as it is learning rather than waiting for the outcome.

**U:**

Upper Bound: A value that is greater than or equal to every element in a set or data.

1. **Conclusion**

In a typical ludo game, *the number of successful defenses is probably greater than the number of successful attacks*, as inferred through this report. The proof-outline involves evaluating different types of piece configurations for their probabilities of successful attacks and defenses and concluding that the configurations with a higher *p*(*Defense*) are the more favorable outcome compared to *p*(*attack*), where *p* is a function indicating the strategy applied.

Without considering the assumptions and rules, the state space complexity of ludo is 5716 ~ 1028. However, with the rules and assumptions the upper bound is and the lower bound is *f*(*no* − 4, *nc*).

The minimum number of expected moves taken to move a distance d is,

Therefore, for one piece to cover a distance of 57 spaces, =16.3 moves, so a player will need a minimum of 260 moves.

There are 2 ways for an expected winner to minimize his/her moves (Ep[m]):

* Follow a defensive strategy to limit the number of moves of their own pieces.
* Follow an aggressive strategy and increase other players’ minimum number of moves, by knocking their pieces off.