Received 1 February 2024, accepted 12 March 2024, date of publication 27 March 2024, date of current version 3 April 2024.

Digital Object Identifier 10.1109/ACCESS.2024.3381958

RESEARCH ARTICLE

Implementation of Alpha-Beta Pruning and Transposition Tables on Checkers Game

CRISTIAN C. SUANCHA[1], **MARCO J. SUAREZ**[1], **AND FELIPE A. BESOAIN**[2], **(Member, IEEE)**

[1]Ingeniería de Sistemas y Computación, Universidad Pedagógica y Tecnológica, Sogamoso 152211, Colombia

[2]Department of Interactive Visualization and VR, Faculty of Engineering, Universidad de Talca, Campus Talca, Talca 3460000, Chile

Corresponding author: Marco J. Suarez (marco.suarez@uptc.edu.co)

This work was supported in part by Proyecto Sistema de Gestión de Investigación (SGI) under Grant 3727, and in part by Universidad Pedagógica y Tecnológica de Colombia (UPTC).

ABSTRACT Checkers is a strategy game for two players on an 8 × 8 square board. This document outlines the creation of a checkers game, employing alpha-beta pruning as a search algorithm and transposition tables to enable the game to learn from past play sessions. The utilization of these techniques enhances the game's strategic decision-making process, contributing to an improved overall gaming experience. The execution time of the machine's moves in games with and without using transposition tables showed that it is possible to make some games faster by using the data stored in the tables. However, under some conditions this can be more time-consuming than the search performed by the alpha-beta algorithm. A user survey resulted in positive outcomes, indicating a favorable user experience when using the implemented game with the transposition tables. Comparative analysis against other algorithms demonstrated the successful implementation of the approach. This research contributes to the field of game AI optimization by integrating alpha-beta pruning and transposition tables into a web-based checkers game. The implementation offers a balance between computational efficiency and user engagement in the web-based checkers game.

INDEX TERMS Checkers, computer games, game algorithm, user experience.

I. INTRODUCTION

Checkers, also known as draughts, is a player vs. player or player vs. machine game based on strategy. This game has been popular for centuries in Anglophone countries. The goal of checkers is to defeat the opponent by "eating" all their pieces, or by making them unable to move any other piece [1]. Zachariah et al. [2] define checkers as a two-player game with a board size of 8 × 8. Only the dark squares of the checkered board are used. Each player moves a piece diagonally to an adjacent unoccupied square at a time. If the adjacent square contains an opponent's piece, and the square immediately beyond it is vacant, the piece may be captured (and removed from the game) by jumping over it [3]. Although there are many rule variations, in almost all variants the player without pieces remaining or who cannot move due to being blocked loses the game [3].

With the advancement of computer technology, it is now possible to develop a computer program that can play checkers at a high level. One approach to developing such a program is to use a combination of a production system and the alpha-beta pruning algorithm [4].

Alpha-beta pruning is a search algorithm used in artificial intelligence to find the best move in two-player strategy games like checkers [5]. The algorithm uses two values, alpha and beta, to limit the number of possibilities that need to be evaluated. A production system, on the other hand, uses a set of rules to make decisions and control the flow of the program [6].

In this context, a potential issue to address is the scalability and efficiency of the search algorithm as the complexity of the checkers game increases. The search space expands exponentially as the game board size and branching factor grow, leading to longer computation times and higher resource consumption. Ensuring that the algorithm remains effective and responsive even for larger game scenarios is a crucial aspect to work on. In this context, while transposition tables help avoid redundant evaluations, their management and memory usage could pose challenges in web-based environments, where resources are limited [7] optimizing the

The associate editor coordinating the review of this manuscript and approving it for publication was Andrea Bottino.

use of transposition tables and exploring ways to minimize memory overhead are vital tasks for improving the overall performance of the implementation [8].

Moreover, web checkers users expect to gain significant knowledge from the implementation of the game. Players will experience faster and more strategic gameplay as the pruning algorithm enables the AI opponent to make smarter moves, enhancing the overall gaming experience. Moreover, this work can serve as a valuable educational resource for students and developers interested in learning about game AI and algorithmic optimizations.

Although production systems alone can give the checkers game developer a summary of how the game works and how it can be implemented in a computer system, including an Alpha-Beta algorithm can also provide developers with the best set of moves that can be implemented so that the computer can defeat real-world players [9]. It is also worth noting that the alpha-beta pruning algorithm has the potential to be improved by implementing transposition tables.

This work describes the development of a game of checkers by using alpha-beta pruning as a search algorithm, and transposition tables to make the game learn from previous play sessions. The article is structured as follows: first, we describe the Alpha beta and transposition tables optimization technique; second, we present the materials and methods, which are principally the main features of the software; third, describe the results and discussion; and finally, we describe our conclusions and future work, including some recommendations based on the experience of the development.

A. ALPHA BETA AND TRANSPOSITION TABLES

Alpha-Beta Pruning is an optimization technique used to reduce the search space of minimax, which is a tree-based algorithm [5]. Minimax is used in sequential games (games where players alternate turns) to determine the most promising move at each game state with the assumption that the opponent plays optimally, and that the available information is organized in a tree-search. As the name of the algorithm suggests, there is a minimizer (the opponent) and a maximizer (the player). The children of max-nodes (maximizers) are min-nodes (minimizers) and vice versa. The depth for a node n_i is the number of edges between n_i and the root, which in this context serves as the number of moves from the current game state to the node. Every game state has a value that is obtained from an evaluation function. An evaluation function receives a board state as input and returns a value as output. Thus, given a depth value, the AI will pick a move that leads to the highest board state value in d turns assuming optimal responses [10].

The tree-search generated for any strategy game with the values of the deeper nodes can explain how minimax tree search is generated with each evaluation value (see Figure 1. Tree-search from minimax algorithm).

As previously stated, checkers is a complex game based on various rules where playing and defeating their opponent

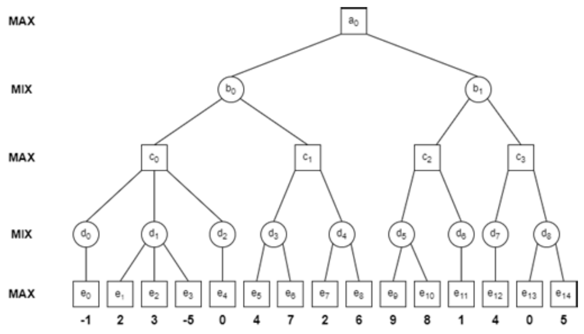


FIGURE 1. Tree-search from minimax algorithm. Adapted from: [3].

gives players happiness and satisfaction. These rules open many possibilities on scientific computation research ground, specifically on Artificial Intelligence (AI) research. AI can make the game more interesting because it can be played using a computer or other electronic devices.

Game theory is a mathematical theory used to analyze strategic interactions among rational decision-makers. It focuses on understanding and predicting the behavior of agents in situations where the outcome depends on the choices made by all participants involved. At its core, game theory seeks to identify Nash equilibria, which are sets of strategies where no player has an incentive to unilaterally deviate from their chosen strategy, given the strategies chosen by the other players. These equilibria represent stable outcomes in strategic interactions, providing insights into decision-making processes and potential outcomes in various scenarios [22].

In this context, when strategy games use artificial intelligence, game theory takes an important role on designing techniques to take any game as a big number of possibilities, in this case, AI development possibilities. As Gupta et al. [11] say, with efficient and accurate game theory and AI algorithms, it is now possible to solve games involving a decision-making method with a huge set of possibilities. Checkers is a zero-sum game, which means that each player's benefit or loss in value is exactly accounted for by the losses or gains in utility of the other players, with a game-tree complexity of about 1040 [11], [12].

In the context of checkers, the rules and conditions of the game determine the possible moves, meanwhile the alpha-beta pruning algorithm can be used to evaluate the best move based on these rules [4]. By combining these two approaches, it is possible to develop a computer program that can play checkers at a high level and make informed decisions based on the rules of the game.

A record is maintained for each leaf node in the tree, documenting the maximum value encountered at each layer. As the search algorithm progresses through the tree, it only seeks to identify the maximum value for each layer before terminating the exploration and moving on to the next layer. This optimized approach enables the algorithm to bypass

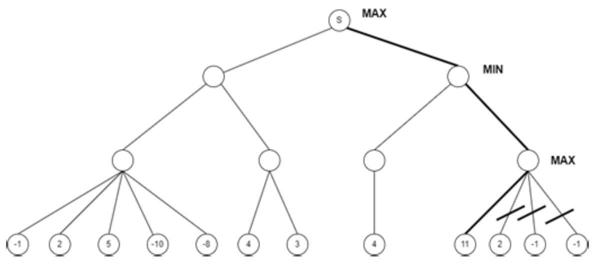


FIGURE 2. Alpha-beta pruning cutting branches.

the exhaustive exploration of all leaf nodes at certain levels, leading to substantial time savings [3].

As shown in Figure 2. Alpha-Beta Pruning cutting branches it is possible to see the pruning of a search tree using the alpha-beta algorithm. It is not necessary to evaluate all the branches of the tree If a good move is found, the algorithm just cuts the other branches and returns the solution.

Also, as the alpha-beta pruning algorithm can be considered as a tree and as a direct acyclic graph [4], moves already visited can be saved in Transposition tables. These tables can reduce the evaluation steps of alpha-beta pruning trees by storing all the moves that were evaluated in the past, saving time when searching the best move in a game. The algorithm searches for an existing move in transposition tables and uses it without evaluating a new tree.

As [8] said, the result of these two cases (tree search and transposition tables) is the same, so we should store information about each move and its score on the transposition table and search there first. Only if that procedure fails, the nodes should be searched.

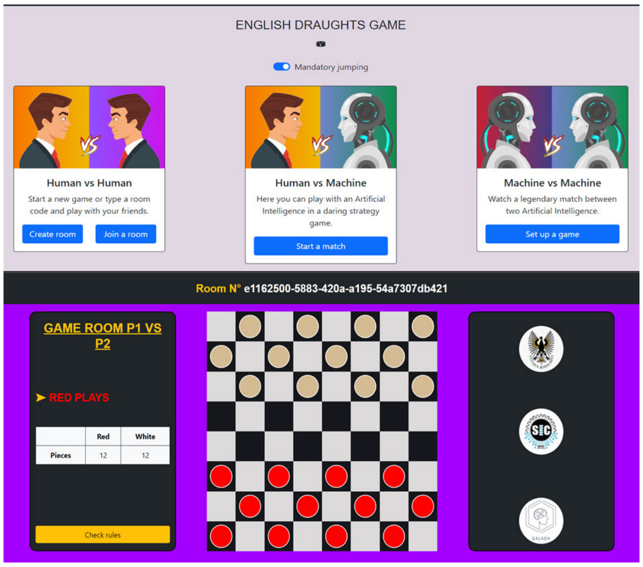
With transposition tables, the alpha-beta pruning algorithm can be improved and made to spend less time when searching for the best move.

This research aims to explore the integration of alpha-beta pruning and transposition tables into a checkers game to enhance strategic decision-making and performance. Significantly, this research addresses scalability challenges in complex game scenarios while maintaining user engagement.

The research question to be addressed in this work is: will the implementation of checkers with alpha beta pruning and transposition tables will optimize performance without losing attractiveness for users? To answer this question, a web version of checkers with the implementation of both algorithms and analyze performance was developed, in addition, tested the software with users to assess the user experience trough a survey.

II. MATERIALS AND METHODS

To create the system, a play session performed by one of the authors and the machine was used. The game followed standard rules, with the addition of mandatory jumping and promotion of pieces to queen status [13]. A modular approach to coding, which leverages Object-Oriented Programming (OOP) principles, was employed. This methodology involves



Start a new game or type a room code and play with your friends.

Here you can play with an Artificial Intelligence in a daring strategy game.

Watch a legendary match between two Artificial Intelligence.

Room N° e1162500-5883-420a-a195-54a7307db421

	Red	White
Pieces	12	12

FIGURE 3. GUI checkers.

the organization of code into distinct functions and modules, facilitating enhanced maintainability and ease of modification [14]. Object-Oriented Programming allow to encapsulate data and behaviour into objects, promoting code reusability and fostering a clearer understanding of the system's structure. To evaluate the best moves for the AI, the alpha-beta pruning algorithm was used as the control strategy, with utility heuristics to assess the value of each state [15].

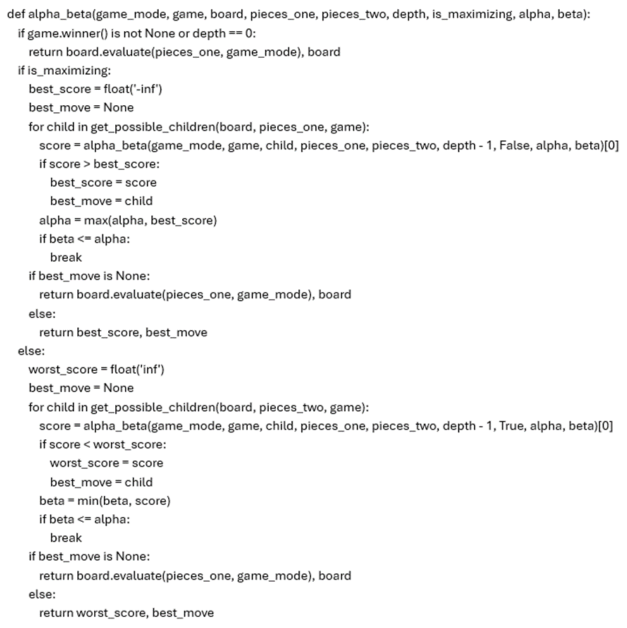
A. DEVELOPMENT

The web game is a distributed system [16], where the frontend calls endpoints for interaction with the game through restful architecture using communication protocols such as fetch and web sockets. The backend is allocated in a different server, but they work as a whole. Various programming techniques were implemented to make the game as playable as possible in a web platform. For example, sockets over specific ports [15], were necessary to create rooms where two people were able to play the game without being interrupted by other people. Frameworks to connect to the backend and some libraries to create synchronization in the game were also used.

Checkers is a game based on strategy [18], which means that "each player is in a singular position where they know nearly everything about their current state" [19]. If a player can know their current state, so can a machine. By understanding the rules and strategies of the game, the machine can use this knowledge to develop a winning strategy and make the best possible moves, in this case, condition statements were used to evaluate how a player can move in the game and how the machine can simulate it.

B. STRATEGY

The possible objects that can simulate checkers were identified. In this way, they created two main objects: pieces, and a board [18]. Each board is created with 12 pieces for



```
def alpha_beta(game_mode, game, board, pieces_one, pieces_two, depth, is_maximizing, alpha, beta):
  if game.winner() is not None or depth == 0:
    return board.evaluate(pieces_one, game_mode), board
  if is_maximizing:
    best_score = float('-inf')
    best_move = None
    for child in get_possible_children(board, pieces_one, game):
      score = alpha_beta(game_mode, game, child, pieces_one, pieces_two, depth - 1, False, alpha, beta)[0]
      if score > best_score:
        best_score = score
        best_move = child
      alpha = max(alpha, best_score)
      if beta <= alpha:
        break
    if best_move is None:
      return board.evaluate(pieces_one, game_mode), board
    else:
      return best_score, best_move
  else:
    worst_score = float('inf')
    best_move = None
    for child in get_possible_children(board, pieces_two, game):
      score = alpha_beta(game_mode, game, child, pieces_one, pieces_two, depth - 1, True, alpha, beta)[0]
      if score < worst_score:
        worst_score = score
        best_move = child
      beta = min(beta, score)
      if beta <= alpha:
        break
    if best_move is None:
      return board.evaluate(pieces_one, game_mode), board
    else:
      return worst_score, best_move
```

FIGURE 4. **Alpha beta pruning method.**

each player in the game. The board evaluates all the possible movements a piece can make, so it is easy to simulate the real game.

Also, as in checkers it is important to know if a piece can become a queen, a piece with a crown was used to represent a queen. On Figure 3. GUI checkers the Graphical User Interface can be seen.

The following algorithmic rules were implemented, to make the game as real as possible:

- A player may not move the same piece to the same two places more than twice in a row.
- Whoever runs out of pieces on the board loses, and the last one is that a player with very few pieces can withdraw from the game, which is done by exiting from the current room.
- Detect when the game has finished. This is achieved by evaluating the possible moves available to each player on each turn and determining if any of them lead to the capture of all the opponent's pieces or to the impossibility of making any move. In this way, it is ensured that the game ends appropriately and results can be displayed.

C. ALPHA-BETA PRUNING

To implement the algorithm, it was necessary to create two specific methods. One of them evaluates the heuristics of each node of the tree [20], which means setting a specific number of utilities that improve the play style of the machine. The second method defines the alpha-beta pruning implementation, evaluating the usefulness of each node, and pruning the branches when it finds that,

$$Alpha \geq Beta. \tag{1}$$

which means that it breaks the execution of the recursive method (see Figure 4. Alpha Beta pruning method).

This algorithm was implemented because it is the best way to simulate that the machine is playing in an intelligent way, as Mohamad Achmad said [1], the application of artificial intelligence in checkers makes the game playable anywhere and anytime. Alpha-Beta pruning is an optimization technique from the Minimax algorithm that can reduce the number of branch/node extensions to get better search results in less time.

The alpha_beta function operates recursively [22], exploring every potential move of the current player. When the is_maximizing function is true, the algorithm seeks to maximize the heuristic value at the current depth level. By contrast, when is_maximizing is false, the algorithm aims to minimize the heuristic.

The heuristic is computed by the evaluation method of the board object, which assesses the current board state and returns a value indicating the favorability of the position for the current player. A random value was introduced to the utility calculation to prevent the game from entering repetitive loops where pieces move to the same spaces repeatedly. The heuristic is computed by the evaluate method of the board object. This function evaluates the current state of the board and returns a value indicating how favorable the position is for the current player. Sometimes the game comes into a loop, moving pieces to the same spaces many times. To avoid that problem, it was necessary to add a certain random value to the utility.

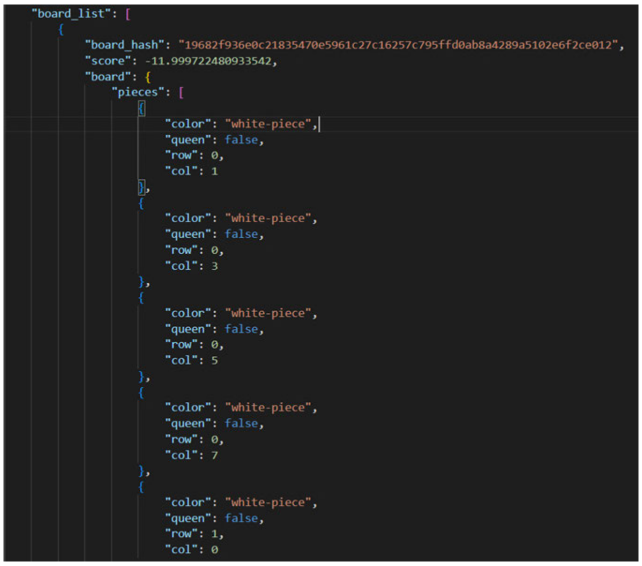
At each depth level, the algorithm updates the alpha and beta values and uses alpha-beta pruning to remove unpromising (child) nodes. If the current value of the evaluation is greater than beta (in the case of the maximizing player) or less than alpha (in the case of the minimizing player), the current branch is pruned, and the algorithm proceeds to the next move. Finally, the function returns the best evaluation found and the corresponding move.

Also, in a comparative study between Monte-Carlo tree search (MCTS) and alpha-beta pruning algorithms, Hikari Kato [9] concludes that the latter method is effective and that "it looks like there is more to gain in playing strength by improving the evaluation function and using classical methods like Alpha-Beta pruning than by increasing the number of playouts using the MCTS strategy".

D. TRANSPOSITION TABLES

Transposition tables were implemented using hashed keys for each board. All the movements made by the machine are saved in a JSON file, along with the hashed key and the evaluation value. Those persisting movements are important for later implementations of AI [8], [23] specifically on alpha-beta pruning algorithm for the game, and on the simulation of learning using the transposition tables (see Figure 5. Transposition table example)

Considering the web checkers implemented, the following parameters can be extracted:



```
"board_list": [
    {
        "board_hash": "19682f936e0c21835470e5961c27c16257c795ffd0ab8a4289a5102e6f2ce012",
        "score": -11.999722480933542,
        "board": {
            "pieces": [
                {
                    "color": "white-piece",
                    "queen": false,
                    "row": 0,
                    "col": 1
                },
                {
                    "color": "white-piece",
                    "queen": false,
                    "row": 0,
                    "col": 3
                },
                {
                    "color": "white-piece",
                    "queen": false,
                    "row": 0,
                    "col": 5
                },
                {
                    "color": "white-piece",
                    "queen": false,
                    "row": 0,
                    "col": 7
                },
                {
                    "color": "white-piece",
                    "queen": false,
                    "row": 1,
                    "col": 0
```

FIGURE 5. Transposition table example, an example of persisting boards.

- Data structure: A matrix sized 8*8 spaces.
- Initial state:
  - 12 pieces for player 1 and 12 for player 2, distributed in 3 rows, each piece separated by a column. Player 1 gets "red" pieces and player 2 gets "beige" pieces.
  - Piece distribution for player 1: [(7,0), (7,2), (7,4), (7,6), (6,1), (6,3), (6,5), (6,7), (5,0), (5,2), (5,4), (5,6)].
  - Piece distribution for player 2: [(0,1), (0,3), (0,5), (0,7), (1, 0), (1,2), (1,4), (1,6), (2,1), (2,3), (2,5), (2,7)].

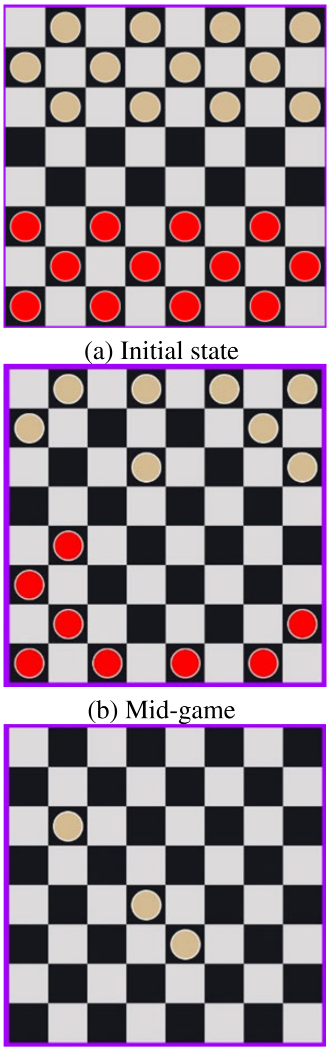
The initial configuration can be seen in Figure 6a. Initial state.

The final state of the game corresponds to a finished match, in which there can be a winner or a draw. A simulation of a match is carried out where there are several moves from each player, until a winner can be determined. Some rules determine the final states of the game, which means there are many ways to finish the game.

The game rules are:

- The player who eats all the opponent's pieces wins;
- if player 1 and player 2's pieces are equal to 1, it is a draw;
- if players do not have any moves remaining, the player who has more pieces wins;
- if players do not have any legal moves available and both players have the same number of pieces, it is a draw.

Another important aspect to define is the space configuration for states. A checkers game has $5 * 10^{20}$ "possible figure positions which, although pale in comparison to chess with approximately 1043 possible figure positions, was a high enough number that the game was not simply solvable and yet small enough for researchers to do their research on" [24].



(a) Initial state

(b) Mid-game

(c) Final state

FIGURE 6. Game states.

Considering the last statement, it can be somewhat difficult to completely define the space configuration of states for this game. However, it is possible to describe the sequence the game follows from initial state, mid-game, and final state (see Figure 6a. Initial state, Figure 6b. Mid-game, Figure 6c. Final state).

Furthermore, inside the space configuration of states for the checkers game, two possible situations can be represented as follows:

- A player makes an illegal movement, where they try to move a piece to another space where there is another piece, or where they try to move a piece two or more spaces than what is allowed. When a movement is illegal, the game shows the possible legal movements a player can make.
- When any player gets to the opposite side of the board, the piece they are using becomes a queen. When this happens, said piece changes to one with a crown (see Figure 7. A piece becoming a queen).

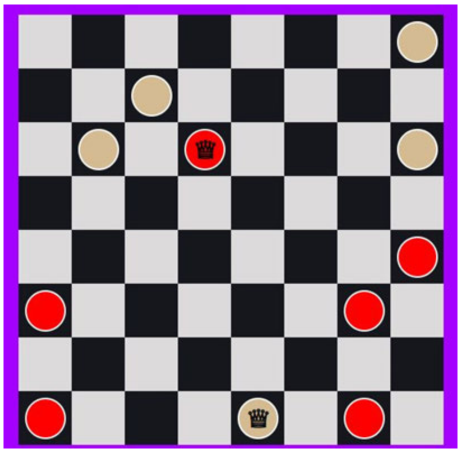


FIGURE 7. A piece becoming a queen.

Transposition tables along with the alpha-beta pruning algorithm are widely used techniques in game programming and artificial intelligence to improve the efficiency of game search algorithms.

The alpha-beta pruning algorithm is a tree search technique that reduces the number of evaluated positions in a game, thus saving computation time. The algorithm is an extension of the minimax algorithm, which evaluates all possible moves in a game tree to determine the best move to make. However, the minimax algorithm is computationally expensive, especially for games with large search spaces. The alpha-beta pruning algorithm seeks to reduce the number of positions evaluated by cutting branches of the search tree that are unlikely to lead to a better outcome.

Transposition tables, on the other hand, are a caching technique used to store previously evaluated game positions and their evaluation values, to avoid reevaluating the same position in the future. When using a transposition table, each game position is associated to a unique key, which is used as an index in a search table. If a position is found in the table, its stored evaluation value can be used. Otherwise, the position is evaluated and stored in the table with its evaluation value.

III. RESULTS AND DISCUSSION

A. IMPLEMENTATION ALPHA-BETA AND TRANSPORTATION TABLES

By combining transposition tables with alpha-beta pruning, game search algorithms can significantly reduce computation time, allowing for more efficient and faster game play [4], [25].

In this study, transposition tables were implemented as a persistence technique to make the machine learn from every move, which is saved on the tables and later used for other gameplays. If the machine finds a play state that is the same as the current one, it just uses the move from the transposition table and does not execute the alpha-beta pruning, reducing the time and machine resources needed to make a move.

The win rate before transposition tables for red and white pieces is approximately the same, while the draw rate stays around 30 games. There is not a clear difference between the two pieces; they are winning with a similar distribution. It is important to mention that they are not using any other search or learning algorithm besides alpha-beta pruning. Those results can be seen at table 1.

TABLE 1. Win rate results of pieces before implementing transposition tables.

Player	Win count
Red	49
White	53
Draw	29

TABLE 2. Win rate results of pieces after implementing transposition tables.

Player	Win count
Red	60
White	58
Draw	37

Results show that white pieces reduced their win rate. As red pieces are using transposition tables, this result suggests that red is learning, but in this case, time is being evaluated, so it is not possible to conclude anything about the win rate. It is important to say that alpha-beta pruning is only implemented for red pieces. Those results can be shown at table 2.

Finally, the average game time was evaluated. The blue line shows the time spent on the game by the player when using transposition tables. This time is sometimes reduced to nearly zero seconds, which means that transposition tables are working as expected. However, the last thirty games sometimes took a long time to finish. This can be due to a slow interaction by the user or to the amount of data the game must read to use the information on the transposition tables (see Figure 8. Average game time).

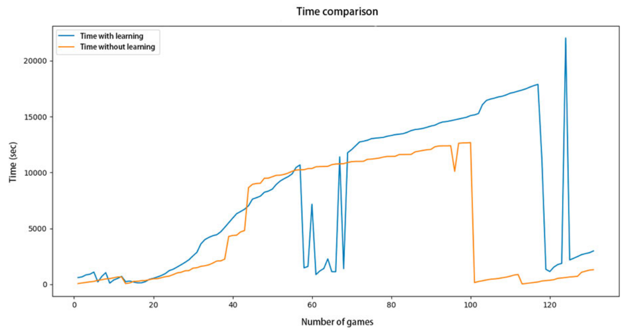


FIGURE 8. Average game time.

It is important to consider that while transpose tables can improve game efficiency by reducing the number of redundant computations, they may also require careful memory management and performance optimisation to avoid significant delays, especially when handling large volumes of data. Implementing strategies such as adaptive table sizing and periodic cleanup of stale entries can help maintain a balance between speed and memory usage.

At the initiation of these measurements, there were no saved games. Following 131 plays involving mandatory captures when jumping over a game piece, and another 131 plays where such captures were optional, a total of 2,796 moves were recorded in the database for the first game mode, while 4,904 moves were recorded for the second mode. The variance in the number of moves can be attributed to the selection process: when captures are not obligatory, the algorithm chooses the most advantageous moves in real-time, whereas in situations with mandatory captures, certain moves are pre-stored for immediate use.

The next step was comparing the implemented checkers game to other existing implementations. This study focused on assessing unique features, and overall user experience in relation to other checkers games. Factors such as AI strategies, search algorithms, performance optimization, and user interface design were evaluated.

Tao et al. [3] used neural networks and specific game rules to create a version of the English checkers game. Their implementation incorporated alpha-beta pruning to enhance gameplay performance, which allowed for best decision-making capabilities. However, the one presented in this document intends to leverage the advantages of a web-based platform, enabling users to enjoy the game across various devices and operating systems. Additionally, utilization of transposition tables optimizes the game's execution times, enhancing overall gameplay performance and providing a foundation for potential integration with neural networks in future iterations. By combining these elements, the implementation described in this document offers both accessibility and scalability, providing an enriched user experience while paving the way for future advancements in game AI and neural network integration.

In a study made by Idzham et al. [7], they developed a web-based English checkers game that incorporates alpha-beta pruning and specific game rules. Their implementation aimed to enhance gameplay performance by efficiently exploring different move possibilities and providing optimal strategies. The game presented in the present document incorporates an additional enhancement by using transposition tables. This improved execution times and established a foundation for future scalability, including the potential integration of neural networks. This improvement further enhances the overall gameplay experience by optimizing move calculations and ensuring efficient decision-making processes.

As seen in both cases, the checkers game developed in this study focuses on the user experience and accessibility by building the game on a web platform and using transposition tables to make the game more scalable and its features easier to perform.

B. USER STUDY

A total of 30 users volunteered to take part in a exploratory study. The survey administered for this study encompassed a diverse array of participants. Ensuring a comprehensive evaluation of the checkers game implementation by including individuals with no prior knowledge of computing, hailing from various academic majors and possessing varying levels of experience. This broad and inclusive approach, allowed to gather insights and feedback that are representative of a diverse user base, ensuring a more holistic assessment of the system's usability and accessibility [26].

TABLE 3. Win rate results of pieces after implementing transposition tables.

Question	Mean	Median	std
How easy was it for you to navigate the English checkers website?	4.5	5.0	0.77
Do you find the website interface intuitive and easy to understand?	4.5	5.0	0.62
How easy was it to find the different functions and features of the checkers game?	4.4	5.0	0.73
How satisfied are you with the visual appearance of the website?	4.4	4.0	0.56
How clear and organized do you find the website design?	4.6	5.0	0.49
Did the instructions and tutorials provided on the website help you understand how to play the checkers game?	4.3	4.0	0.53

Participants were instructed to engage with each of the three distinct checkers game modes on a single occasion, after which they were requested to assess their experiences via the survey. This approach ensured that the evaluation process was based on a one-time interaction with each game mode, providing a snapshot of user feedback and impressions in a controlled way.

The survey included questions that aimed to assess the user 1) experience of the web-based platform, 2) level of satisfaction with the visual appearance of the website, 3) usefulness and 4) intention. In this context, Participants were requested to rate the level of ease they encountered while navigating the website, their perception of the interface's intuitiveness and clarity, as well as how easy it was to locate the different functions and features within the game [27]. Responses were collected using a 5-point likert scale [28], where higher scores represented greater comfort or ease.

The outcome of the survey demonstrated that participants had an overall positive experience with the website (see Table 3). 5, indicating a high level of comfort while exploring the site. This suggests that the website's layout and organization effectively facilitated seamless movement across various sections and pages of the game.

Overall, the results indicate a high level of user satisfaction with the web platform for playing checkers. Participants found the website easy to navigate, the interface intuitive and comprehensible, and the different functions and features of the game conveniently accessible. These positive findings underscore the effectiveness of the implemented design choices and highlight the overall user-friendliness of the web platform.

The survey included additional questions that focused on assessing the usability of the web platform. Participants were asked to rate their level of satisfaction with the visual appearance of the website, their perception of the clarity and organization of the design, and the helpfulness of the instructions and tutorials provided to understand the game. Responses were measured on a scale from 1 to 5, where 5 indicated a high level of satisfaction.

The results of the survey showed that participants experienced high levels of satisfaction with the usability of the website. When evaluating the visual appearance of the site, most respondents rated their satisfaction with a score of 4 or 5, indicating that they were happy with how the website looked like. This suggests that the visual design elements were well received by users, contributing to a better experience.

Participants also reported positive perceptions for the other questions. This indicates that the layout and presentation of the information on the website were effective in providing a seamless and user-friendly experience. The answers demonstrated that participants were highly satisfied with the usability of the web platform for playing English checkers. Users expressed contentment with the visual appearance of the website, perceived the design as clear and organized, and found the provided instructions and tutorials helpful. These positive results reflect the successful implementation of user-friendly features and contribute to an overall positive user experience on the web platform.

Following the assessment of usability, participants were also asked to report if they encountered any noticeable problems or delays while interacting with the website.

The survey results indicated that most respondents reported no issues with performance or speed on the web platform. This positive outcome suggests that the website was able to deliver a smooth and responsive experience to users, allowing them to seamlessly navigate and interact with the checkers game. Participants' feedback indicated that they did not encounter any significant hindrances or frustrations due to performance-related factors. Only two responses (6,7%) reported issues during game execution, and only one said that the website was *"too slow"*.

Then, participants were asked to evaluate if the website adapted properly to their devices and screen size. This question aimed to assess the responsiveness and compatibility of the web platform across different devices and screen resolutions.

Most respondents reported positive experiences regarding the website's adaptation to their devices and screen sizes. Participants expressed satisfaction with the responsive design, noting that the website adjusted seamlessly to their specific device, whether it was a desktop computer, laptop, tablet, or smartphone. This suggests that the web platform successfully catered to various screen sizes, providing an optimized and visually pleasing experience for users.

Only three responses (10%) manifested issues with responsiveness features, and some of them said that *"on mobile devices, the entire dashboard is not visible, it becomes tedious to scroll to see the entire dashboard"* and *"the game stopped giving me options with the white tiles once, despite having options for that within the game"*. Those problems could be related to the users' internet connection or to other issues out of the hands of the developers.

TABLE 4. Win rate results of pieces after implementing transposition tables.

Question	Yes	No
Did you experience any performance or speed issues while using the website?	2	28
Did the website adapt correctly to your device and screen size?	27	3
Would you recommend the English checkers game website to other individuals interested in playing checkers?	29	1

The survey concluded with the question "would you recommend the English checkers game website to other individuals interested in playing checkers?". This inquiry aimed to assess the level of satisfaction participants had towards the website.

The results of the survey revealed that most respondents expressed a positive inclination towards recommending the website to others. They indicated a willingness to endorse the English checkers game website based on their own positive experiences and satisfaction with the platform. This suggests that users found the website to be engaging, enjoyable, and well-suited for playing checkers, which underscores the perceived quality of the gaming experience provided and the level of confidence they had in the platform. The users' approval attests to the success of the implemented techniques and algorithms in creating an appealing and user-friendly web-based checkers game.

These results can be seen at Table 4.

IV. CONCLUSION AND FUTURE WORK

There is potential to further improve the efficiency and effectiveness of the checkers game search algorithm by incorporating machine learning and deep learning techniques. For example, one could use reinforcement learning to develop a heuristic evaluation function that can more accurately assess the value of a game position [28]. Alternatively, one could use neural networks for the machine to directly learn a policy for making moves, based on a given game position. Additionally, one could explore the use of Monte Carlo tree search techniques, which can improve the efficiency of game search algorithms by sampling likely game outcomes. By incorporating these advanced techniques into the checkers game, it may be possible to create an even more challenging and engaging game that can adapt to the play style of individual users [29].

In addition to machine learning and deep learning techniques, forecasting may also be useful in the context of the checkers game. Specifically, forecasting could be used to predict the likely moves of the opponent based on their past moves and playing style using transposition tables. This could be done by using time-series analysis techniques from transposition tables, which can identify patterns and trends in time-series data, such as the sequence of moves made by the opponent. By forecasting the opponent's likely moves, it may be possible to develop more effective strategies for winning the game, such as anticipating and countering the opponent's moves. However, forecasting in this context may be challenging due to the complexity of the game and the large number of possible moves, and further research is needed to determine the feasibility and effectiveness of this approach.

To further augment the intelligence and sophistication of the web checkers game, the incorporation of neural networks as an additional strategy holds great promise. By integrating neural networks into the decision-making process, the AI opponent can learn from past gameplay experiences and adapt its strategies dynamically. Neural networks offer the potential to capture complex patterns and features of the game, enabling the AI to make more informed and strategic moves using deep learning. This fusion of traditional search algorithms like alpha-beta pruning and transposition tables with the learning capabilities of neural networks can lead to a more challenging and engaging gaming experience for players.

Implementing a checkers game can be a complex task, requiring careful consideration of the game rules and logic. The production of such a system is the best way to become familiar with how a strategy game works, and it also serves as the initial step to implement an AI algorithm. The game was tested and validated through simulations of multiple moves by both players, utilizing system production for validation as well.

Overall, the implementation of a checkers game as a software serves as an example of applying programming concepts and techniques in game and simulation development. Alpha-beta pruning is an effective algorithm for optimizing the game's performance by reducing the number of nodes that need to be evaluated. Other authors referenced in this research paper have also stated that it is the most effective algorithm for this strategy game.

Also, the implementation of transposition tables in the checkers game, along with the alpha-beta pruning algorithm, can significantly improve the efficiency of the game search algorithm. The use of transposition tables allows for the caching of previously evaluated positions, which can save significant computation time by avoiding the reevaluation of the same positions. Additionally, the alpha-beta pruning algorithm reduces the number of positions evaluated by cutting off branches of the search tree that are unlikely to lead to a better outcome.

The combination of transposition tables and alpha-beta pruning is a powerful technique that can improve the performance of game search algorithms, allowing for faster and more efficient gameplay. By implementing these techniques in the English checkers game, it is possible to create a more challenging and engaging game that can be enjoyed by players of all skill levels.

The survey results indicated a high level of user satisfaction with the web platform for the game. Participants reported comfort and ease in navigating the website, finding the interface intuitive and easy to understand, and locating the different functions and features of the game without significant difficulty. The visual look, clarity, and organization of the website design received positive feedback from users. The instructions and tutorials provided were found to be helpful in understanding how to play the checkers game.

Additionally, participants reported no significant issues with performance or speed on the web platform, indicating a smooth and responsive experience. The positive responses demonstrate the successful implementation of the strategies and technologies, including alpha-beta pruning and transposition tables, which contributed to an enjoyable and challenging gaming experience.

Considering positive feedback and the level of user satisfaction, it can be concluded that the developed checkers game is likely to be recommended to other individuals interested in playing checkers. The findings highlight the successful integration of artificial intelligence techniques and algorithms, coupled with a user-friendly interface, resulting in a compelling and engaging gaming experience.

These conclusions provide valuable insights for the continued improvement and development of the web platform, ensuring that it meets the expectations and needs of users interested in playing English checkers.

ACKNOWLEDGMENT

The authors would like to thank the Research Center, Facultad Seccional Sogamoso, Universidad Pedagógica y Tecnológica de Colombia, for their invaluable support and resources, which greatly contributed to the successful completion of this research and also would like to thank the Faculty of Engineering, Talca University, for their guidance and collaboration, enhancing the quality and depth of this work. Their unwavering support is sincerely appreciated.

REFERENCES

[1] M. A. A. Abdullah and H. M. Judi, "Try dam: Digital checkers game application based on machine learning," *Int. J. Academic Res. Bus. Social Sci.*, vol. 13, no. 1, p. 818, Jan. 2023, doi: 10.6007/ijarbss/v13-i1/15561.

[2] A. Zachariah, A. Louis, P. Kumar, G. Balamurali, and A. Martin, "Checkers using reinforcement learning," in *Proc. Int. Colloq. Recent Trends Eng. (IC@MACE)*, Apr. 2020.

[3] J. Tao, G. Wu, and X. Pan, "Design and application of computer games algorithm of checkers," in *Proc. Chin. Control Decis. Conf. (CCDC)*, 2020, pp. 231–234.

[4] A. Parashar, A. K. Jha, and M. Kumar, "Analyzing a chess engine based on alpha–beta pruning, enhanced with iterative deepening," in *Expert Clouds and Applications* (Lecture Notes in Networks and Systems), vol. 444. Singapore: Springer, 2022, pp. 691–700, doi: 10.1007/978-981-19-2500-9_51.

[5] O. Marckel, "Alpha-beta pruning in chess engines," in *Proc. UMM CSci Senior Seminar Conf.*, 2017, pp. 1–6.

[6] S. Mandadi and S. Vijayakumar. (2020). *Implementation of Sequential and Parallel Alpha-Beta Pruning Algorithm View Project*. [Online]. Available: https://www.researchgate.net/publication/343945419

[7] K. K. Idzham, M. W. N. Khalishah, Y. W. Steven, M. S. M. F. Aminuddin, H. N Syawani, A. Zain, and Y. Yusoff, "Study of artificial intelligence into checkers game using HTML and Javascript," *IOP Conf. Ser., Mater. Sci. Eng.*, vol. 864, no. 1, May 2020, Art. no. 012091, doi: 10.1088/1757-899x/864/1/012091.

[8] Z. Qi, X. Huang, Y. Shen, and J. Shi, ''Optimization of Connect6 based on principal variation search and transposition tables algorithms,'' in *Proc. Chin. Control Decis. Conf. (CCDC)*, Aug. 2020, pp. 198–203.

[9] H. Kato, S. Z. Fazekas, M. Takaya, and A. Yamamura, ''Comparative study of Monte-Carlo tree search and alpha-beta pruning in amazons,'' in *Information and Communication Technology: Third IFIP TC 5/8 International Conference, ICT-EurAsia 2015, and 9th IFIP WG 8.9 Working Conference, CONFENIS 2015, Held as Part of WCC 2015, Daejeon, Korea, October 4–7, 2015, Proceedings 3*. Springer, 2015, pp. 139–148.

[10] D. Ye and J. Trossing, ''Playing the fox game with tree search: MCTS vs. alpha-beta,'' Bachelor's Degree Thesis, School Elect. Eng. Comput. Sci. (EECS), Stockholm Univ. Appl. Sci., Stockholm, Sweden, Sep. 2022.

[11] P. Gupta, Vividha, and P. Nagrath, ''Checkers-AI: American checkers game using game theory and artificial intelligence algorithms,'' in *Deep Learning in Gaming and Animations*. Boca Raton, FL, USA: CRC Press, Dec. 2021, pp. 1–18, doi: 10.1201/9781003231530-1.

[12] B. Gill, ''Machine learning in checkers: Using coevolution to drive game strategy,'' Bachelor of Science thesis, Univ. Exeter, Exeter, U.K., Tech. Rep., 2023, doi: 10.13140/RG.2.2.23824.17928.

[13] E. E. Kopets, A. I. Karimov, G. Y. Kolev, L. Scalera, and D. N. Butusov, ''Interactive robot for playing Russian checkers,'' *Robotics*, vol. 9, no. 4, p. 107, Dec. 2020, doi: 10.3390/robotics9040107.

[14] R. Al-Msie'Deen, A. H. Blasi, and M. A. Alsuwaiket, ''Constructing a software requirements specification and design for electronic IT news magazine system,'' *Int. J. Adv. Appl. Sci.*, vol. 8, no. 11, pp. 104–118, Nov. 2021, doi: 10.21833/ijaas.2021.11.014.

[15] S. Slyman, M. Gillies, and V. Lytra, ''Developing an evaluation framework for analysing educational simulation games,'' in *Proc. 16th Eur. Conf. Games Based Learn.*, 2022, vol. 16, no. 1, pp. 526–534, doi: 10.34190/ecgbl.16.1.363.

[16] J. Verbraeken, M. Wolting, J. Katzy, J. Kloppenburg, T. Verbelen, and J. S. Rellermeyer, ''A survey on distributed machine learning,'' *ACM Comput. Surveys*, vol. 53, no. 2, pp. 1–33, Mar. 2020, doi: 10.1145/3377454.

[17] N. Bonelli, F. D. Vigna, A. Fais, G. Lettieri, and G. Procissi, ''Programming socket-independent network functions with nethuns,'' *ACM SIGCOMM Comput. Commun. Rev.*, vol. 52, no. 2, pp. 35–48, Jun. 2022, doi: 10.1145/3544912.3544917.

[18] G. Fabris, L. Scalera, and A. Gasparetto, ''Playing checkers with an intelligent and collaborative robotic system,'' *Robotics*, vol. 13, no. 1, p. 4, Dec. 2023, doi: 10.3390/robotics13010004.

[19] J. Parker, *Game Development Using Python*. Mercury Learning and Information, 2021, Accessed: Feb. 24, 2023.

[20] M. Esteve, J. J. Rodríguez-Sala, J. J. López-Espín, and J. Aparicio, ''Heuristic and backtracking algorithms for improving the performance of efficiency analysis trees,'' *IEEE Access*, vol. 9, pp. 17421–17428, 2021, doi: 10.1109/ACCESS.2021.3054006.

[21] A. N. W. Jofanda and M. Yasin, ''Design of checkers game using alpha-beta pruning algorithm,'' *INTENSIF: Jurnal Ilmiah Penelitian dan Penerapan Teknologi Sistem Informasi*, vol. 5, no. 2, pp. 279–295, Aug. 2021, doi: 10.29407/intensif.v5i2.15863.

[22] T. Bilen and B. Canberk, ''Overcoming 5G ultra-density with game theory: Alpha-beta pruning aided conflict detection,'' *Pervas. Mobile Comput.*, vol. 63, Mar. 2020, Art. no. 101133, doi: 10.1016/j.pmcj.2020.101133.

[23] Y. Xie, W. Gao, Z. Dai, and Y. Li, ''Research and improvement of alpha-beta search algorithm in Gobang,'' *Adv. Transdisciplinary Eng.*, vol. 20, pp. 819–829, Feb. 2022, doi: 10.3233/ATDE220084.

[24] J. Popic, B. Boskovic, and J. Brest, ''Deep learning and the game of checkers,'' *MENDEL*, vol. 27, no. 2, pp. 1–6, Dec. 2021, doi: 10.13164/mendel.2021.2.001.

[25] B. Vollenwyder, S. Petralito, G. H. Iten, F. Brühlmann, K. Opwis, and E. D. Mekler, ''How compliance with web accessibility standards shapes the experiences of users with and without disabilities,'' *Int. J. Hum.-Comput. Stud.*, vol. 170, Feb. 2023, Art. no. 102956, doi: 10.1016/J.IJHCS.2022.102956/BIBTEX.

[26] S. Ma, B. Zhao, Z. Hou, W. Yu, L. Pu, and L. Zhang, ''Robust visual object tracking based on feature channel weighting and game theory,'' *Int. J. Intell. Syst.*, vol. 2023, pp. 1–19, Jul. 2023, doi: 10.1155/2023/6731717.

[27] A. T. Jebb, V. Ng, and L. Tay, ''A review of key Likert scale development advances: 1995–2019,'' *Front. Psychol.*, vol. 12, May 2021, Art. no. 637547, doi: 10.3389/FPSYG.2021.637547.

[28] A. Bashar, ''Survey on evolving deep learning neural network architectures,'' *J. Artif. Intell. Capsule Netw.*, vol. 2019, no. 2, pp. 73–82, Dec. 2019, doi: 10.36548/jaicn.2019.2.003.

[29] G. Tan, P. Wei, Y. He, H. Xu, X. Shi, and P. Yi, ''An algorithm based on valuation forecasting for game tree search,'' *Int. J. Mach. Learn. Cybern.*, vol. 12, no. 4, pp. 1083–1095, Apr. 2021, doi: 10.1007/s13042-020-01222-3.

CRISTIAN C. SUANCHA is currently a Research Assistant with the Systems and Computer Engineering Program, Seccional Sogamoso Faculty, Universidad Pedagógica y Tecnológica de Colombia, demonstrates a keen interest in the fields of artificial intelligence and advanced programming. Engaging actively in research endeavors, their work revolves around exploring innovative applications and methodologies within these domains. With a commitment to contributing to the academic community, the research assistant is dedicated to advancing knowledge in the intersection of systems and computer engineering, particularly emphasizing the cutting-edge realms of AI and advanced programming.

MARCO J. SUAREZ received the M.Sc. degree in information management from the Colombian School of Engineering Julio Garavito, in 2012, and the Ph.D. degree in strategic planning and technology management from UPAEP-Mexico, in 2016. He was with ECI University-Colombia and UPAEP Mexico. He is currently an Associate Professor with UPTC Colombia. He is also a Research Associate with the Ministry of Science, Technology and Innovation, Colombia. His research interests include machine learning, AI, and deep learning. He serves on the editorial board for numerous technology journals.

FELIPE A. BESOAIN (Member, IEEE) received the B.S. degree in bioinformatics engineering from the Faculty of Engineering, Universidad de Talca, Talca, Chile, in 2010, the M.S. degree in free software, in 2012, and the Ph.D. degree in network information technologies from Universitat Oberta de Catalunya, Barcelona, Spain, in 2018. In 2021, he was a Postdoctoral Researcher in social psychology with Universidad de Talca, combining app development with current attitude change theory. He is currently a Researcher in various research and development projects with demonstrable experience in the development of mobile, immersive technologies (virtual reality) in the industry of health, agronomy, tourism, culture, and heritage. His current research interests include the application of persuasive technologies in intelligent contexts for the promotion of health, tourism and cultural heritage, the use of ubiquitous computing and mobile devices, gamification, and immersive technologies for the form/change of attitudes.