Explainable Search for SameGame
Final Report
Semester 2 Year 1

Benedikt Hornig (i6294146), Elliot Doe (i6192244), Lukas Padolevicius (i6167544),Xander Wigman (i6296167)
Group 12

Project Supervisors: Anna Wilbik, Mark Winands
Project Coordinator: Gijs Schoenmakers

***Abstract*—Explaining why intelligent search agents such as Monte Carlo Tree Search (MCTS) chose an action over another is often not easily and intuitively explainable. Moreover, differences in scoring values provided by intelligent search agents do not suffice to explain why an action was chosen over another. Therefore, we present a method of finding relevant features for the given game SameGame played by MCTS and putting these scoring differences into human understandable explanations. Relevancy of a feature was determined by the value of these features compared to other available actions and combining it with offline correlation analysis between the feature values and the scores given by MCTS. Additionally, explaining the moves chosen by MCTS with a decision tree was explored.**

I. Introduction

The explanation behind the reasoning of an AI can give significant inside to users and enhance the trust between the user and the AI. This is important to all use cases in which humans eventually interact with the decision making process in some way, shape, or form. This decision making process can be applied to games, which can offer a low stakes environment in which explanations can be tested and refined. A new emerging focus is on explaining goal-driven systems as opposed to data-driven systems [1]. For this reason the focus of this paper lies on a search tree algorithm, such as Monte Carlo Tree Search, in the simplistic testing environment of single player game, SameGame.

A. *Explainable AI*

Explainable AI (XAI) is an important, constantly expanding, field, whose importance seems to grow as AIs are applied more and more to other fields of research and in everyday life. It is important that the user is able to trust the decisions suggested by the system and it can help developers when improving and debugging [1].

Trust is an especially important factor when discussing an AI's position in the decision making process. It has been shown that humans tend to more heavily hold mistakes made by an algorithm against them than when judging humans. This holds true even if shown that the algorithm is outperforming humans [2]. Trust can be built by answering concerns that may arise from unexpected behaviour from the AI. These concerns are often related questions concerning both the short-term planning and long-term plans of the AI. It can be shown that it is beneficial to offer explanations and that these can quell these concerns to a degree. Often times this is more easily done using an interactive approach where the user can request specific information addressing their concerns [3]. It is suggested that, to decrease the effect of this bias, a clarification can be given, proactively, to avoid surprise. This can be done by offering an explanation beforehand, this could give the user new insight and understanding as to the conclusion that the AI came to [4].

There can be several ways to offer an explanation, the most prevalent and explored method of explaining within XAI focuses on explaining a specific decision. The focus can also be placed on explaining the ranking between the different available choices. This is what *explainable search* attempts to achieve [5]. This offers the ability to compare different rankings to each other and additionally offers user interaction. Allowing for the user to more easily identify biases, errors, and the intent of the model [6].

B. *Research Questions*

To research explainable AI in relation to SameGame and Monte Carlo Tree Search the following questions are formulated:

1) Which features can be extracted from the search algorithms used for SameGame?
2) How do extracted features correlate to the effectiveness of moves they are in?
3) How can extracted features be used to offer an explanation for predicted choices?

II. Use Case

The explainability for single player games is explored within SameGame using Monte Carlo Tree Search (MCTS). MCTS was chosen over a method, like A*, to explore generating explanations for a method that does not produce a fully explored tree, and can thus can be applied more generally.

A. SameGame

The rules of SameGame are relatively simple and fairly important for the purposes of formulating an explanation. As such, the rules of the game are as follows. SameGame is a game in which a game area, of 15x15 tiles, is laid out in front of the player. Within this area there are several differently coloured tiles. Groups of more than two, non-diagonally, adjacent tiles can be select and removed from the game area for points, these groups are called clusters. The points gained scale quadratically with the amount of tiles in a cluster, according to the formula $(n-2)^2$ with n being the size of the group. The game ends when no more moves can be made. Clearing all the tiles is rewarded with 1,000 additional bonus points[7].

Gravity is in place moving tiles downwards when tiles are removed from underneath them. Additionally, if a column is fully removed the all the right tiles of it are shifted to the left to push the remaining tiles together.

B. Monte Carlo Tree Search

In MCTS a search tree is built before deciding on a move. This is done by running Monte Carlo simulations on up to the leaf nodes, with each node representing a specific game state. The scores at each node are the averaged scores of the scores obtained by this node [8].

Within the time allotted to the MCTS run the following steps are followed. First the tree is traversed to a current leaf node from the root node. Secondly random moves are selected until the game is finished from the game state at the selected leaf node. Then thirdly, the first node from the played out game not present in the tree is added to the tree. Lastly, the backpropogation is executed and the average score of nodes are updated, as are the amount of results. After running through this process for the allotted time a final move is selected. This is done by choosing the child with the highest scoring full game [7].

III. Methodology

The explanations are built onto a pre-built MCTS model for SameGame, in Java [7]. The first step was identifying important features that could be used when deciding a move within SameGame. These features had to be both easily interpreted by users and relevant to the game in someway. The focus was placed on human interpretability, this was done by brainstorming to find possible relevant features that could be easily visualised within the game space.

After having identified and selected which features were to be collected, the process for extraction was started. To extract the features from the board state the features were subdivided into categories. These features were then extracted from the board using a Java program, with individual classes for each category. This was done so that the extraction process could be added onto the pre-built Java code [7].

After the extraction of the features, the next step was to analyse the extracted data. A standardized set of 20 games were played and their data was collected.[1] From this data set correlation terms with respect to the score given by MCTS were created, using Pearson correlation coefficients. These correlation scores could then be used as relevance scores when creating the decisions. Allowing for negative relations to be extracted as well.

The explanations were then generated. This phase was split into three steps.

1) The first step was executed during the data analysis phase. This step was to alter the user interface to allow for space to show the explanations. Additionally, the Java code was expanded to detect the move the mouse hovers over, so that the relevant explanation could be shown.
2) The second step was detecting which features were the most relevant for the currently selected move and adding the comparison with the best possible move selected by the MCTS. This was done through a comparison of the value of a feature of this move with the average value of that specific feature of all possible moves, normalising this score and then applying the relevance score, generated in the data analysis step. By using two value thresholds, which were found empirically, the best and worst features of a move can be found.
3) The third step was then adding the human interpretable explanations. These were hard coded in, and offer a sentence that explains why this move was chosen. This is based on the best and worst features found for that move and a comparison with the best move that MCTS selected.

IV. Features

The first step in creating explanations is the identification of human recognisable features. This can aid in the explanations offered to the user, by relating the decision making process of MCTS to features that humans would identify and use when playing SameGame themselves. These features were divided into two categories. The first category, *game features*, are the features relating to the current state of the game, such as the amount of red tiles or the size of the biggest cluster. Whereas, the second category, *tree features*, present possible future moves and can help explain the reason the biggest cluster was not chosen and a smaller cluster was picked instead, for instance, because more points could be scored later on.

A. Game Features

The game features themselves are subdivided into the categories: *cluster features*, *column features*, *move features*, and *playable area features*. Offering features for the different structures and the general layout of the current game state.

1) Cluster Features: For the *cluster features* all clusters currently in the game are stored, together with their individual features. Features such as their colour, size, and position are saved, as shown in Fig. 1. From this the largest cluster can be

[1]The positions can be found at: www.js-games.de/eng/games/samegame

deduced, as well as the singletons which cannot be removed from the board currently. With the size of the clusters directly corresponding to the game score this is a valuable measure to observe.

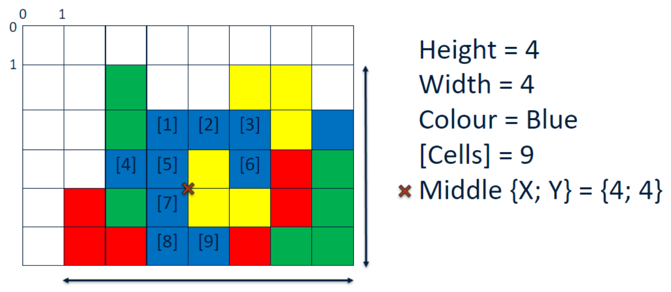


Fig. 1. Example *cluster feature*: Showing the current largest cluster, a Blue cluster consisting of 9 tiles.

2) Column Features: Column features explore the features of the various columns making up the current game state. The various colours that are present and the height of the columns are saved, as shown in Fig. 2. With the removal of a column shifting and reshaping the entire board, this can offer valuable insight into future moves when combined with *tree features*.

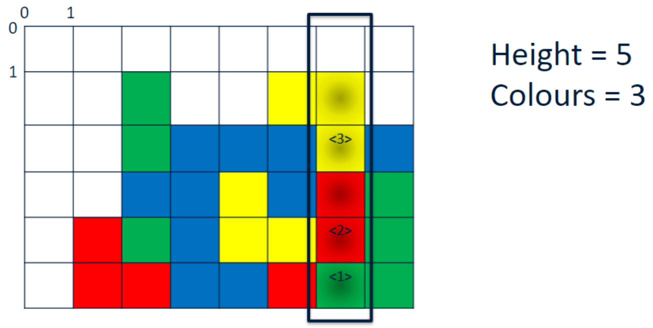


Fig. 2. Example *column feature*: Showing a column, how many colours are present and its height.

3) Move Features: Move features describe the changes that will occur when a specific move is made. Similar to the *cluster features*, these features allow specific explanations relating to specific moves made. Important features stored are the destroyed and gained connections, as shown in Fig. 3.

4) Playable Area Features: The features within the *playable area* category relate to the total current state of the game. The number of empty tiles and the number of tiles of each colour are collected as well as the general shape, as shown in Fig. 4. This can offer insight into the progress of the game so far or if a specific colour is focused to be removed first.

B. Tree Features

The inherent features of the MCTS tree are known as *tree features*. These features are things such as the distance from the root node, the distance to the leaf nodes, or number of

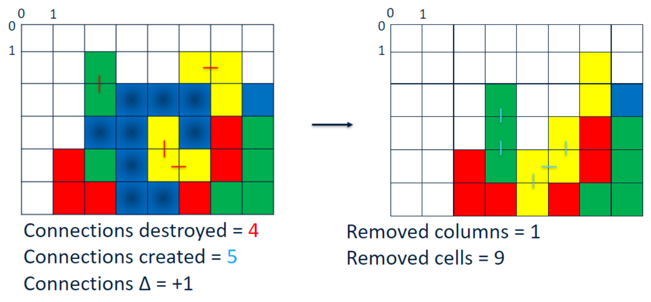


Fig. 3. Example *move feature*: Shows the move made, how many connections destroyed, gained, net connections gained, and the removed cells and columns.

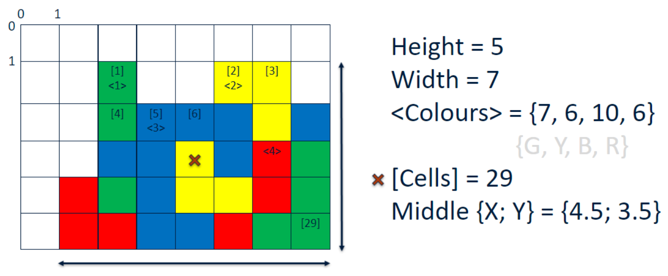


Fig. 4. Example *playable area feature*: Shows the shape, amount of tiles, relative centre, and the colours and their amounts present in the playable area.

children or siblings. Due to time constraints and because these features are generally less easily interpretable by the layman these were left out. The node ID and game step were still collected however. In future research a suggested approach would be linking the *tree features* to different collections of *game features*. This would allow for more a human interpretable explanations using tree features, such as "In 3 moves a cluster of 8 is created." or "This move is bad because it finishes the game too quickly leaving too many tiles.".

C. Feature Extraction

The feature extraction is done on the stored nodes of the tree built by MCTS for one turn. Each category of *game features* are determined for the individual nodes, whereas the *tree features* are detected from the entire tree. All the data is stored in a json file formatting, sorted into different files according to their categorisation.

V. Explainability

The final necessary step of the project is the explanation of the chosen moves. There are various ways in which the features and decisions can be explained, but not all are suitable for a dynamic environment like SameGame. To most clearly describe features, the method should be reactive and within the same application as the game. While offline analysis such as charts or graphs are useful in some cases, they do not respond to the user in any way. To tackle this hurdle, the explanations are instead given in natural language and during the game. The explanations are also reactive to user input. This is achieved by explaining to a user what is good and bad about a move they want to learn more about. By hovering over any move, a user can see the advantages and disadvantages of the move (relative to the other moves) and can also see why the game playing agent chose a different move instead.

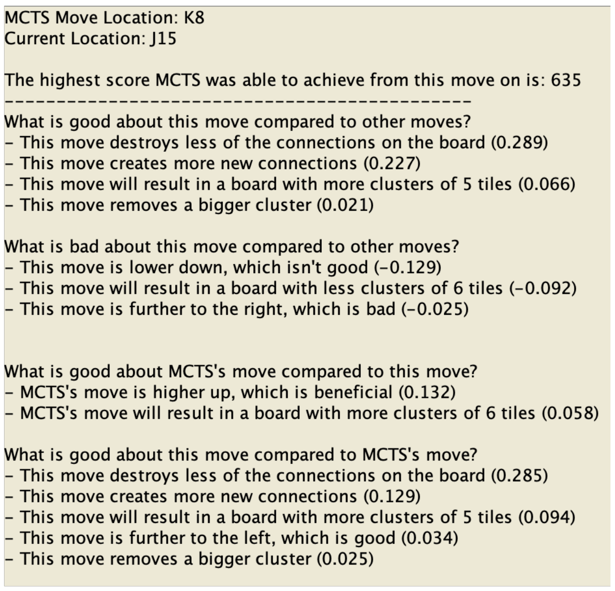
A. Correlation Table

Due to the size of the tree generated by MCTS it was not fully stored for the data analysis later. Instead, for every instance of the MCTS deciding on a best move, a file is created for the tree used to determine this best move. From this tree, the important area is that of the root node, it's direct descendants, and the highest scoring paths of each of those descendants. From this area, every node that has been visited at least 10 times by the MCTS will be included in the final dataset, as a collection of it's features.

By comparing each of these features with their corresponding MCTS score, for every move of each of the 20 benchmark games, each feature can be assigned a global correlation coefficient to be used for building explanations.

TABLE I
CORRELATION COEFFICIENTS OF EACH FEATURE TO THE MCTS SCORE.

Feature	Correlation Coefficient
MCTS Score	1.00
Number of Removed Cells	0.11
Number of Removed Columns	-0.25
Colour	-0.015
Connections Destroyed	0.2
Connections Created	0.25
Move Column	-0.13
Move Height	-0.18
Average Cluster Size	0.35
Largest Cluster	0.51
Clusters of size 2	0.065
Clusters of size 3	0.068
Clusters of size 4	0.066
Clusters of size 5	0.12
Clusters of size 6	0.21
Clusters of size 7+	0.49
Average column height	0.31
Highest column	0.35
Average colours per column	0.35



MCTS Move Location: K8
Current Location: J15

The highest score MCTS was able to achieve from this move on is: 635

What is good about this move compared to other moves?
- This move destroys less of the connections on the board (0.289)
- This move creates more new connections (0.227)
- This move will result in a board with more clusters of 5 tiles (0.066)
- This move removes a bigger cluster (0.021)

What is bad about this move compared to other moves?
- This move is lower down, which isn't good (-0.129)
- This move will result in a board with less clusters of 6 tiles (-0.092)
- This move is further to the right, which is bad (-0.025)

What is good about MCTS's move compared to this move?
- MCTS's move is higher up, which is beneficial (0.132)
- MCTS's move will result in a board with more clusters of 6 tiles (0.058)

What is good about this move compared to MCTS's move?
- This move destroys less of the connections on the board (0.285)
- This move creates more new connections (0.129)
- This move will result in a board with more clusters of 5 tiles (0.094)
- This move is further to the left, which is good (0.034)
- This move removes a bigger cluster (0.025)

Fig. 5. Live explanations of MCTS scores

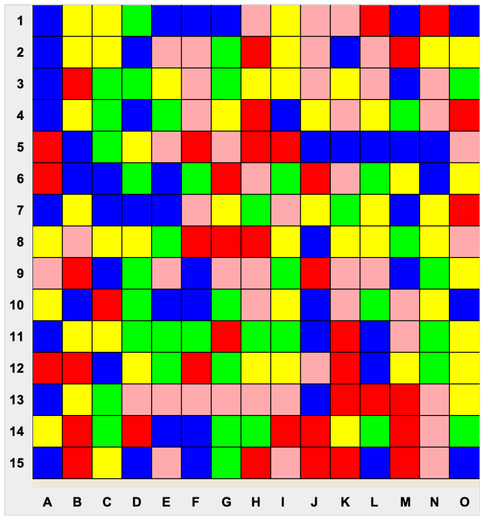


Fig. 6. Board corresponding to the given explanations in figure 5

B. Implementing Explanations

To generate the explanations, as shown in Fig. 5, a link must be made between the data acquired from the correlation coefficients and the real time information of the current game. This paper proposes the use of a relevance table. The correlation coefficient table can be generated offline, be saved to the game, and finally be used to generate explanations. This final step is the most difficult, as there are many different ways to accomplish this. This paper proposes the following method: First the value of each feature is found. This value is extracted from the game state that results from the move which the user is considering. The value of a feature by itself is not very meaningful, thus instead the deviation from the norm is considered. This is because a user needs to know why to pick a feature *instead of another*. Finally, this deviation is then multiplied by the feature correlations in the correlation table. This results in the relevance of each feature. Relevance here does not necessarily mean relevance to the game or even to the score, but instead the relevance to the decision of choosing a certain move. All of this together results in the following formula for the relevance score for each feature i:

$$S^i_{relevance} = (F^i_{selected} - F^i_{avg})/(F^i_{avg} + 1) * C^i_{corr} \quad (1)$$

Where $F^i_{selected}$ is the value of the ith feature in the currently selected move, F^i_{avg} is the average value of the ith feature, and C^i_{corr} is the correlation coefficient from the table. This formula makes it so that all scores given to features are normalised and also keeps negative correlation into account. The lack of a bad feature could be an equally important explanation as the abundance of good ones.

To make a decision on which values to include a threshold was introduced. For the good features related to a move any features with a score above 0.015 were used. Similarly, any features with a score below -0.015 were used for the bad features. These thresholds were found empirically. These values can be adjusted for a stricter or more general explanations.

The explanations themselves are then shown to the left of the game window. The explanations that are given are:

1) What is good about the selected move compared to others?
2) What is bad about the selcted move compared to others?
3) What is good about MCTS's move compared to the selected move?
4) What is good about the selected move compared to MCTS's move?

C. Decision Tree Based Explanations

In addition to the user-interactive explanations discussed prior, another way to generate explanations is to fit a model that is inherently explanatory using the game features to predict the MCTS score. This is currently being done post-game, using the playouts by the MCTS to fit a regression decision tree (e.g. Fig. 7).

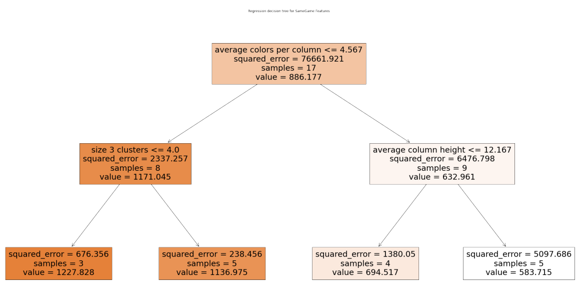


Fig. 7. Example of a decision tree using SameGame features

The scope of the data used by the tree can vary, and currently two different trees have been examined. One of the trees, which will be referred to as a 'global' tree, fits the data of 20 known benchmarks games for SameGame. The other tree type will be referred to as a 'local' tree, which models a single game.

1) Global Decision Tree: The global decision tree, included in full in Appendix A, uses 17155 instances , excludes root nodes, and is slightly pruned by requiring 10 instances to make a rule and split a branch. Deciding when to split a branch is based on mean squared error, and it's accuracy is represented by a coefficient of determination of $R^2 = 0.634$.

An example of an explanation for a user move would consist of traversing the tree using the features of that move, each branching during this traversal giving an explained decision in the form of a rule. A example of this traversal is shown in Tab. II.

In addition, the decision tree also estimates the importance of different features, in Fig. 8, these features are represented by order of their importance.

TABLE II
SET OF RULES DENOTING THE PATH TAKEN IN THE GLOBAL DECISION TREE BY AN ARBITRARY SAMEGAME MOVE
SCORE GIVEN BY MCTS = 1661.55
SCORE PREDICTED BY MODEL = 1079.22

Order	Rule	Feature Value
1	largest cluster <= 8.50	6
2	average colours per column >1.70	4.8
3	numRemovedCells <= 52.00	4
4	average colours per column >4.70	4.8
5	size 7+ clusters <= 4.50	0
6	size 2 clusters <= 24.50	18
7	average colours per column <= 4.83	4.8
8	size 4 clusters >3.50	6
9	average column height <= 14.03	12.6
10	size 2 clusters >16.50	18
11	size 7+ clusters <= 2.50	0

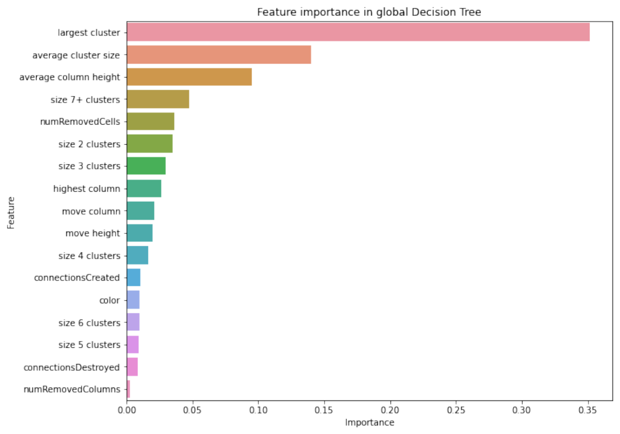


Fig. 8. Feature importances in the global decision tree

2) Local Decision Tree: For local decision trees, the process is the same as global decision trees, except that the data used is reduced to only one of the benchmark games, reducing the amount of instances to 965. This smaller tree has a coefficient of determination of $R^2 = 0.498$, which is slightly lower than it was for global decision trees.

However, although the accuracy only had a slight change, the features that the model deems important have changed dramatically. Fig. 8 deemed the size of the largest cluster to be by far the most important, followed by average cluster size and average column height further down, with highest column being a relatively unimportant feature. But, looking at Fig. 9, suddenly the highest column becomes the most important feature, closely followed by average column height, and the 2 most important features in the global tree become much more meaningless.

This development would suggest that an arbitrary game of SameGame may have more specific preferences on which features are important when modelling the MCTS prediction, since the features have changed so dramatically, yet the accuracy did not change nearly as dramatically. However, since the accuracy did decrease for the local tree, it likely overfit the data to an extent, and since the accuracy did not

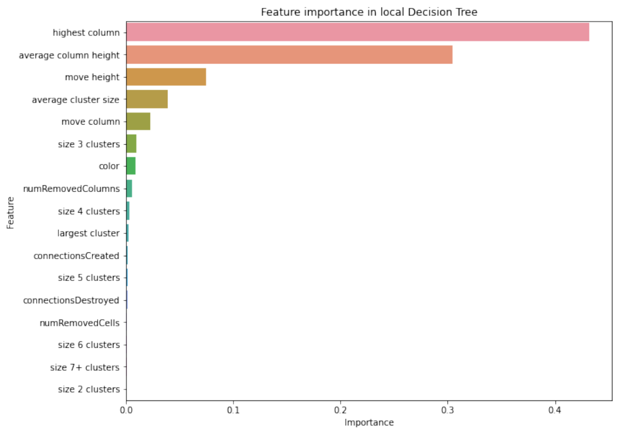


Fig. 9. Feature importances in the local decision tree

change too much, it also means the global tree was either underfitting the data, which seems unlikely since the expansion was not heavily pruned, overfit the entire dataset, or that the SameGame is very difficult to model accurately based on the current set of features.

VI. Conclusion

This paper proposed one way to tackle XAI in SameGame. The proposed method consists of feature extraction, offline data analysis and finally online interactive explanations. The final product is an MCTS implementation of SameGame that can be interrupted and explored through user interactive means. Any move can be explored and feedback will be given as to why that move is good or bad. Moves are also compared to the move MCTS would choose to provide insight into the final decision MCTS made. Finally a decision tree is provided to offer additional insight into MCTS offline.

VII. Discussion

XAI is a very complex field of study. This project only looked into the explainability of MCTS in one very specific domain. This explanation alone constituted a whole project worth of work. This goes to show how difficult general XAI could potentially be. For this project a lot of domain knowledge and user input was required to properly explain the agent. Things such as the considered features to the data analysis were all work that is not easily scaled to other domains or projects.

On top of the difficulty of generalising the approach to XAI, there are many different features that can be chosen. This project attempted to find as many relevant features as possible, but it might well be that many more features exist in SameGame that the authors were not aware of. Even when looking at just the features that were extracted, there are a lot of ways to consider the features. This project mostly considered the correlation between one feature and one score, but there might be hidden synergies or interactions between features that were not spotted. Not only are these interactions difficult to find, they are difficult to explain.

Another difficulty that was encountered is that the accuracy of the explanations are difficult to measure. However, it can be argued that the accuracy can be measured through user studies. The inherent accuracy to what the MCTS chose and why is not necessarily as relevant when constructing the explanation as is the belief of the user that the explanation was accurate. This can be seen when for instance looking at an example move where the MCTS predicts that a certain move is good because three moves down the line it sees that a cluster of size 5 can be chosen. This, however, can still change as the MCTS generates a new tree after every move. Say that on the second move it discovers a new path that it sees as much better, this would not change its previous decision, but it would change what the previous explanation predicted. This would make the more accurate explanation, thus seem less accurate to the user and to the decisions in reality made. It can thus be seen that the accuracy of an explanation has two factors: the actual taken path (the human view) and the predicted path at the time of the decision (the machine view). More research would have to be done into which factor is the most relevant, however, the explanations are there to build trust in the agent. As such the human view would be the suggested view with more importance.

The final difficulty of working with XAI is the subjective nature of the field. The effectiveness of an explanation varies from person to person, which makes studying them difficult. User studies can be done but are difficult and time-consuming which limits the effectiveness and speed of development in XAI. This project was unfortunately not able to do the user studies, and might be considered for future work.

References

[1] H. Baier and M. Kaisers, "Towards explainable mcts," in *2021 AAAI Workshop on Explainable Agency in AI*, 2020.

[2] B. J. Dietvorst, J. P. Simmons, and C. Massey, "Algorithm aversion: people erroneously avoid algorithms after seeing them err.," *Journal of Experimental Psychology: General*, vol. 144, no. 1, p. 114, 2015.

[3] H. Baier and M. Kaisers, "Explainable search," in *2020 IJCAI-PRICAI Workshop on Explainable Artificial Intelligence*, 2020.

[4] M. T. Gervasio, K. L. Myers, E. Yeh, and B. Adkins, "Explanation to avert surprise.," in *IUI Workshops*, vol. 2068, 2018.

[5] A. Dey, C. Radhakrishna, N. N. Lima, S. Shashidhar, S. Polley, M. Thiel, and A. Nürnberger, "Evaluating reliability in explainable search," in *2021 IEEE 2nd International Conference on Human-Machine Systems (ICHMS)*, pp. 1–4, IEEE, 2021.

[6] J. Singh and A. Anand, "Exs: Explainable search using local model agnostic interpretability," in *Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining*, pp. 770–773, 2019.

[7] M. P. Schadd, M. H. Winands, M. J. Tak, and J. W. Uiterwijk, "Single-player monte-carlo tree search for samegame," *Knowledge-Based Systems*, vol. 34, pp. 3–11, 2012.

[8] R. Coulom, "Efficient Selectivity and Backup Operators in Monte-Carlo Tree Search," in *International Conference on Computers and Games*, pp. 72–83, Springer, 2006.

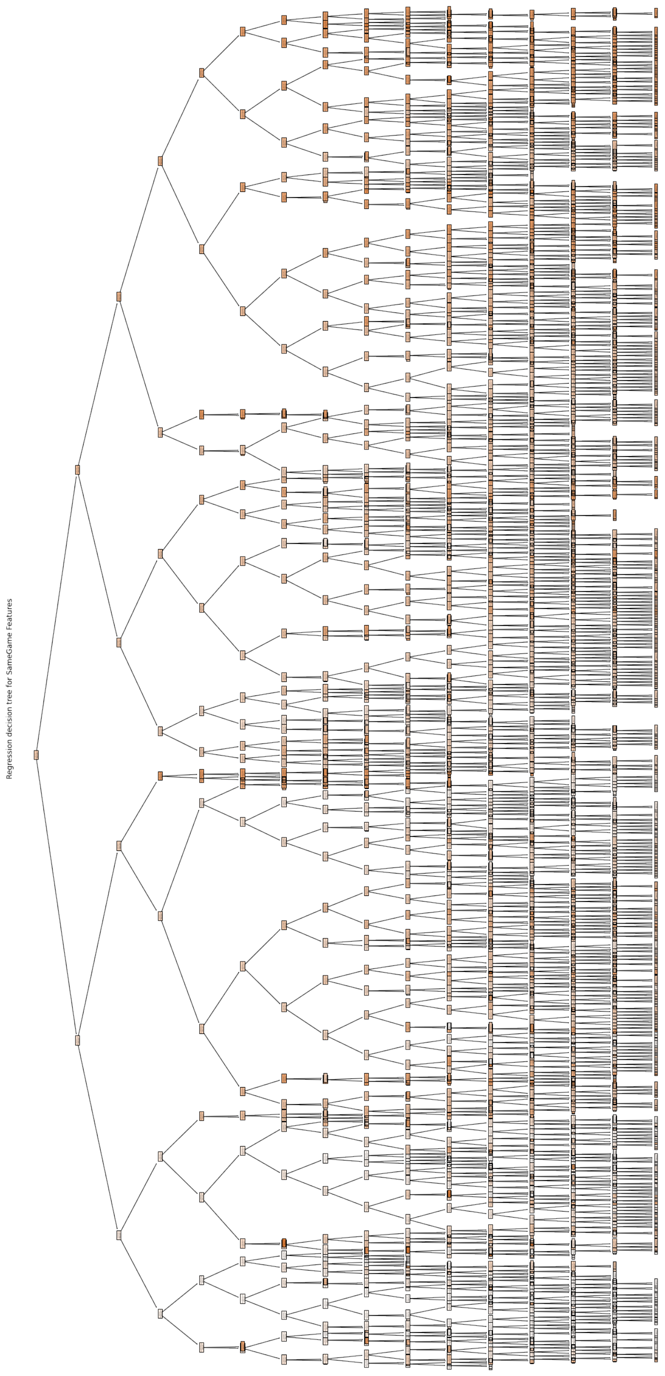


Fig. 10. Global tree.

Appendix
Global Decision Tree