**A COMPREHENSIVE REVIEW ON ARTIFICIAL INTELLIGENCE BASED**

**MACHINE LEARNING TECHNIQUES FOR DESIGNING INTERACTIVE CHARACTERS**

* Paper reviews current literature based on the challenges in real time strategy game and explores the tasks in a real-time environment.
* To achieve optimal performance many techniques have been used along with the neural networks, which are prepared end to end in tasks such as object recognition video games and board games.
* The comparative analysis is made on different AI based machine learning techniques to comprehend the performance issues and other challenges faced while trying to obtain optimal performance.
* Paper contains discussion on the advantages and disadvantages of each technique to understand and compare the efficiencies with the existing techniques based on the strategies.
* Techniques used for review of the artificial intelligence-based machine learning techniques for designing interactive characters naive Bayes classifier, support vector machines, genetic algorithms and neural networks.
* The review techniques were used to create a new intelligent, serious game and to provide the players with a real experience.
* These techniques were compared and it was observed that each technique has its own pros and cons whereas neural networks are better compared to other techniques and thus help in designing of the player characters in video games

[P.S – waste paper, has grammatical errors in it -\_-]

**EFFICIENT REINFORCEMENT LEARNING IN ADVERSARIAL GAMES**

* Algorithm: MiniMax
* Paper presents a learning approach based on the Least-Squares Policy Iteration (LSPI) algorithm that overcomes MiniMax optimality criterion or under optimization limitations by focusing on learning a state-action evaluation function.
* The approach to learning in a variety of two-player, turn-taking, tactical adversarial games (Eg: Backgammon, Othello/Reversi, Chess, Hex, etc) consists of updating some state evaluation function usually in a Temporal Difference (TD) sense either under the MiniMax optimality criterion or under optimization against a specific opponent.
* Limitations of MinMax:
  + updates to the evaluation function are incremental
  + stored samples from past games cannot be utilized
  + quality of each update depends on the current evaluation function due to bootstrapping.
* Advantage of LSPI : agent can make batch updates to the evaluation function with any collection of samples, can utilize samples from past games, and can make updates that do not depend on the current evaluation function since there is no bootstrapping.
* Things from paper that we can use:
  + For any given state [board, player, parity] of the game, we compute the following:
    - mobility: Number of available moves for the player in the current board.
    - stability: Number of player’s discs in the current board whose color cannot change in the rest of the game.
    - frontier: Number of player’s discs in the current board adjacent to empty squares.
    - square(i, j): Content (player disc, opponent disc, or empty) of square (i, j) in the current board.
* Experimental results and graphs for Mukesh

**EVALUATING A REINFORCEMENT LEARNING ALGORITHM WITH A GENERAL INTELLIGENCE TEST**

* Paper applies recent notion of anytime universal intelligence tests to the evaluation of a popular reinforcement learning algorithm, Q-learning.
* Show that a general approach to intelligence evaluation of AI algorithms is feasible.
* Simple terms: Evaluates progress of AI, that is, does performance and intelligence test.
* Evaluation is better when the problem or task is understood completely (more time understanding the nuts and bolts)
* Paper uses “notion of universal distribution and the related algorithmic information theory” to “define a universal distribution of tasks for a given AI realm, and sort them according to their (objective) complexity.”

(Personally, I do not know what this means)

* The experiment run in this paper portrays a view about how an implementation of the intelligence test using the environment class can be used to evaluate AI systems.
* The goal of the paper was not to analyse some well-known properties of Q- learning (such as convergence, state overloading, etc.) or to designate a `winning' algorithm. The goal of the paper, rather, was to show that a top-down (theory-derived) approach for evaluating AI agents can work in practice.

**GENERAL VIDEO GAME EVALUATION USING RELATIVE ALGORITHM PERFORMANCE PROFILES**

* Paper explores the idea of characterising game quality through playing a game with different controllers and comparing performance.
* It does so by investigating the relative performance of different general game-playing algorithms.
* Seven game-playing algorithms was used to play several hand-designed, mutated and randomly generated VGDL game descriptions.
* The seven controllers were used to play through a set of example-, mutated and randomly generated games.
* Result show that win rates suggest a relationship between intelligent controllers' success and better game design; for better designed games, the relative performance of different types of algorithms differ more.
* Average wins for generated, mutated and example games are graphically represented
* Paper hypothesised that the performance difference between good and bad game-playing algorithms is higher on well-designed games, and therefore can be used as at least a partial proxy for game quality.
* The results corroborate the paper’s initial conjecture, showing a clear distinction between results of more and less intelligent controllers for human-designed games but not for random games.
* Images from game for explanation

**REAL-TIME CONNECT 4 GAME USING ARTIFICIAL INTELLIGENCE\***

* Problem statement: The study presented a design that converted connect 4 game into a real-time game by incorporating time restraints.
* Approach: The design used Artificial Intelligence (AI) in implementing the connect 4 game. The AI for this game was based on influence mapping.
* Results: A waterfall-based AI software was developed for a Connect 4 game.
* Conclusion: A real time connect 4 game was successfully designed and implanted with GUI using C++ programming language.
* The concept of the Connect 4 game is to get, before your opponent, four chips in a row, arranged either diagonally, vertically, or horizontally.
* Algorithms considered are minimax, minimax with alpha-beta pruning, A\* and influence maps.
* Time was used to check performance
* Multiplayer
* Explanation and flowchart

**HEURISTIC SEARCH WHEN TIME MATTERS**

* Paper propose an approach – Bugsy. It incorporates the utility function directly into the search, obviating the need for a separate termination policy.
* It is based on off-line parameter tuning and a novel benchmark domain for planning under time pressure based on platform-style video games.
* Bugsy does not require any off-line training.
* First paper to apply anytime monitoring to anytime heuristic search.
* Paper presents a very simple portfolio-based method that estimates a good parameter to use for a bounded-suboptimal search algorithm to optimize a given utility function.
* Bugsy is a best-first search algorithm that accounts for the user’s preference between search time and solution cost.
* Difference between Bugsy and most other methods for trading-off deliberation time and solution cost is that Bugsy considers the trade-off directly in the search algorithm, whereas previous techniques, such as those based on anytime algorithms, only consider the trade-off externally to the actual search algorithm.
* Background research on heuristic and suboptimal searches and utility functions are done for the paper
* Paper is huge, has 4 pages worth of references.

**LARGE-SCALE OPTIMIZATION FOR EVALUATION FUNCTIONS WITH**

**MINIMAX SEARCH**

* Paper presents Minimax Tree Optimization (MMTO), to learn a heuristic evaluation function of a practical alpha-beta search program.
* MMTO ensures the existence of a local minimum within a convenient range of parameters.
* Paper demonstrates the performance of MMTO in shogi, a variant of chess where evaluation functions need to handle a wider variety of features and positions than in Western chess.
* Minimax Tree Optimization (MMTO) is an extension of comparison training to reach the first intuitive goal. The purpose of this extension is to overcome the practical difficulties and stabilize the mathematical optimization procedure with a largescale feature weight vector ‘w’.
* Given a set of training positions ‘P’ and the desired move ‘dp’ for each position ‘p’, MMTO optimizes the weight vector ‘w’ so that the minimax search with w better agrees with the desired moves.
* Paper has search depth and iteratively searches and update using partial derivatives.
* Bonanza was used for the experiments, it uses techniques such as MMTO, PVS, a capture search at frontier nodes as a quiescence search, transposition tables, static exchange evaluation, killer and history heuristics, null move pruning, futility pruning, and late move reductions.
* MMTO consists of two procedures:
  + a shallow heuristic search for all training positions using the current feature weights.
  + an update guided by an approximation of the gradient of the objective function.
* MMTO outperformed the existing methods and the experimental results on the rate of agreement and playing strength indicate that MMTO can adjust forty million parameters.

**MCTS-MINIMAX HYBRIDS**

* Monte-Carlo Tree Search (MCTS) is a sampling /based search algorithm that is state of the art in a variety of games.
* MC samples moves instead of considering all legal moves from a given state, which allows it to handle large search spaces with high branching factors. It also uses Monte-Carlo simulations that make it independent of a static heuristic evaluation function to compare non-terminal states.
* The Monte-Carlo rollouts of entire games gives it a strategic advantage over traditional depth-limited minimax search with pruning.
* These MC rollouts can often detect long-term consequences of moves, freeing the programmer from having to capture these consequences in a heuristic evaluation function.
* MCTS runs a higher risk than full-width minimax search of missing individual moves and falling into traps in tactical situations, because of its higher selective tree.
* Because MCTS builds a highly selective search tree, focusing only on the most promising lines of play, it has been conjectured that it could be less appropriate than traditional, non-selective minimax search in domains containing a large number of terminal states and shallow traps.
* Shallow traps are features of domains that are problematic for MCTS, in particular Chess.
* Paper proposes MCTS-minimax hybrids that integrate shallow minimax searches into the MCTS framework. It explores ways of combining the strategic strength of MCTS and the tactical strength of minimax in order to produce more universally useful hybrid search algorithms.
* Three approaches are outlined, using minimax in the selection/expansion phase, the rollout phase, and the backpropagation phase of MCTS.
* These hybrid algorithms are a first step towards combining the strategic strength of MCTS and the tactical strength of minimax.
* Paper investigates the algorithm’s effectiveness in the test domains of Connect-4, Breakthrough, Othello, and Catch the Lion, and relates this performance to the tacticality of the domains.
* MC works by repeating the following four-phase loop until computation time runs out. The root node of the tree represents the current state of the game. Each iteration of the loop represents one simulated game.
  + Selection
  + Expansion
  + Rollout
  + Backpropagation
* Paper describe three different approaches for applying minimax with alpha beta pruning within the MCTS framework.
  + Minimax in the Rollout Phase – MCTS-MR
  + Minimax in the Selection and Expansion Phases – MCTS-MS
  + Minimax in the Backpropagation Phase – MCTS-MB
* In all experimental conditions run in the paper, the results of hybrids were compared against regular MCTS-Solver as the baseline.
* The proposed variant MCTS-MS significantly outperformed regular MCTS with the MCTS-Solver tension in Catch the Lion, Breakthrough, and Connect-4.
* The same holds for the proposed MCTS-MB variant in Catch the Lion and Breakthrough, while the effect in Connect-4 is neither significantly positive nor negative.
* MCTS-MR, was quite strong in Catch the Lion and Connect-4 but significantly weaker than the baseline in Breakthrough, suggesting it might be less robust with regard to differences between domains such as the average branching factor.
* Figures

**A SURVEY OF MONTE CARLO TREE SEARCH METHODS\***

* Monte Carlo Tree Search (MCTS) is a search method that combines the precision of tree search with the generality of random sampling, it is a method for finding optimal decisions in a given domain by taking random samples in the decision space and building a search tree according to the results.
* Paper surveys literature to the date, it intends to provide a snapshot of the state of the art after the first five years of MCTS research. Paper outlines the core algorithm’s derivation, imparts some structure on the many variations and enhancements that have been proposed, and summarise the results from the key game and non-game domains to which MCTS methods have been applied.
* The basic MCTS process is conceptually very simple. A ‘tree’ is built in an incremental and asymmetric manner.
* For each iteration of the algorithm, a ‘tree policy’ is used to find the most urgent node of the current tree.
* The tree policy attempts to balance considerations of exploration (look in areas that have not been well sampled yet) and exploitation (look in areas which appear to be promising).
* A ‘simulation’ is then run from the selected node and the search tree updated according to the result. This involves the addition of a child node corresponding to the action taken from the selected node, and an update of the statistics of its ancestors.
* Moves are made during this simulation according to some ‘default policy’, which in the simplest case is to make uniform random moves.
* Benefit of MCTS - the values of intermediate states do not have to be evaluated, as for depth-limited minimax search, which greatly reduces the amount of domain knowledge required. Only the value of the terminal state at the end of each simulation is required.
* Game theory extends decision theory to situations in which multiple agents interact. A game can be defined as a set of established rules that allows the interaction of one or more players to produce specified outcomes.
* Games are classified by the following properties:
  + Zero-sum: Whether the reward to all players sums to zero (in the two-player case, whether players are in strict competition with each other).
  + Information: Whether the state of the game is fully or partially observable to the players.
  + Determinism: Whether chance factors play a part (also known as completeness, i.e. uncertainty over rewards).
  + Sequential: Whether actions are applied sequentially or simultaneously.
  + Discrete: Whether actions are discrete or applied in real-time.
* Games are typically modelled as trees of decisions as follows:
  + Minimax attempts to minimise the opponent’s maximum reward at each state, and is the traditional search approach for two-player combinatorial games. The search is typically stopped prematurely and a value function used to estimate the outcome of the game, and the alpha-beta heuristic is typically used to prune the tree. The maxn algorithm is the analogue of minimax for non-zero-sum games and/or games with more than two players.
  + Expectimax generalises minimax to stochastic games in which the transitions from state to state are probabilistic. The value of a chance node is the sum of its children weighted by their probabilities, otherwise the search is identical to maxn. Pruning strategies are harder due to the effect of chance nodes.
  + Miximax is similar to single-player expectimax and is used primarily in games of imperfect information. It uses a predefined opponent strategy to treat opponent decision nodes as chance nodes.
* MCTS Algorithm:

General MCTS approach.

**function** MCTSSEARCH(s0)

create root node v0 with state s0

**while** within computational budget do

vl <- TREEPOLICY(v0)

x <- DEFAULTPOLICY(s(vl))

BACKUP(vl,x)

**return** a(BESTCHILD(v0))

* UCT is used to solve the determinized games sampled independently by HOP.
* Information Set UCT (ISUCT)
* Sparse UCT
* UCT+

[lots of data – great paper]

**MINIMAX GUIDED REINFORCEMENT LEARNING FOR TURN-BASED STRATEGY GAMES**

* Games are a good medium for artificial intelligence (AI) research, since they compare user and machine behaviour directly.
* Core AI tasks in games is to find a better solution of a problem, developing an AI for games requires that a viable solution should be found at limited time and space.
* Paper investigates the use of learning which is derived from dynamic scripting to provide action in a turn-based strategy game.
* Proposed algorithm is based on the automatic rule ordering of dynamic scripting, but it combined static rule list with dynamic rule ordering through reinforcement learning.
* The algorithm is then combined with the Minimax algorithm to achieve a better performance.
* The performance of the proposed algorithm is evaluated through a series of matches against a static manually designed AI.
* The result shows that the proposed algorithm is able to adapt the static AI at a shallow Minimax depth.
* The algorithm also shows the ability reducing calculation time and using less memory space. It is concluded that Minimax guided reinforcement learning can be applied to the turn-based strategy genre.
* The proposed algorithm is able to defeat the static AI by a percentage of 76.8%. The proposed algorithm also achieved a better result than the modified Minimax algorithm.
* Even though it is proven that Minimax guided reinforcement learning is capable of adapting to static AI, it is yet to be seen whether or not the AI can stand on itself against a human player.

[Action rules have been mentioned]

**ON MONTE CARLO TREE SEARCH AND REINFORCEMENT LEARNING**

* Paper’s shows that a straightforward adaptation of RL semantics within tree search can lead to a wealth of new algorithms, for which the traditional MCTS is only one of the variants.
* Paper confirms that planning methods inspired by RL in conjunction with online search demonstrate encouraging results on several classic board games and in arcade video game competitions. The study promotes a unified view of learning, planning, and search.
* The relationship between the MCTS and RL is somewhat opaque, with most applied researchers being unaware of the connections between both the methods. One of the main factors for this lack of cross fertilization is the absence of a common language between dedicated game researchers and RL researchers.
* Paper outlines the reasons for the separate evolution of the two communities and comprehensively analyse the connection between the fields of MCTS and RL. Focusing
* on the game AI community by starting from the MCTS point-of-view, the article identifies both the similarities and the differences between the two fields.
* Paper thoroughly describe the MCTS mechanics with RL theory and analyse which of them are unusual for traditional RL methods and can thus be understood as novel.
* To demonstrate how RL mechanics can be beneficial also in MCTS like algorithms, article develops the temporal-difference tree search (TDTS) framework, which can be understood both as a generalization of MCTS with temporal-difference (TD) learning, as well as an extension of traditional TD learning methods with novel MCTS concepts.
* It showcases a TDTS algorithm, named Sarsa-UCT, where we
  + successfully apply a bootstrapping learning method to the original MCTS framework when using an incrementally-growing tree structure with a tabular, non-approximated representation, and without domain-specific features;
  + efficiently combine it with an upper confidence bounds (UCB) selection policy
  + evaluate and analyse TD backups under such conditions.