# AI310/CS361 ARTIFICIAL INTELLIGENCE FALL 2025

## COURSE PROJECT INSTRUCTIONS

**Instructions to Students:**

- This is a group work project. Each group consists of **six students** (“Mainstream Programme”) or **four to six** (“both Software Engineering & Medical Informatics Programmes”). *The Teaching Assistant must approve group members through registration*. Each group must develop the assigned idea using Python.
    - **Project Objectives:** The objectives of this project can be summarised as applying the main ideas, fundamental concepts, and basic algorithms in the fields of artificial intelligence and machine learning.
- **Submission:** Submission is done according to the following schedule:
    - **Week 12/13: Submission and Discussion of the** (1) **<u>Project</u>** and (2) **<u>Documentation</u>**. *The report should include the following: (1) Project idea in detail, (2) Main functionalities, (3) Similar applications in the market, (4) A literature review of Academic publications (papers) relevant to the idea (at least 4 to 6 papers, as per the number of team members), (5) the Dataset employed (preferably a publicly available dataset), (6) Details of the algorithm(s)/approach(es) used and the results of the experiments, and (7) Development platform.*
- **Assessment:** Assessment will be on the reports, code submitted, and discussions with team members. All team members must contribute across all phases, and each member's role must be clearly stated in each report.
    - The Project will be assessed based on the following criteria:
        - The complexity of the problem, & the correctness of the algorithms employed.
        - The quality/comprehensiveness of your experiments & documentation.
        - The correctness of your analysis and design diagrams.
        - Implementation correctness.
- **Feedback:** If requested, further details and feedback could be provided for each group through discussions with the teaching assistant(s) during the weekly-labs/office-hours.
- You can only submit your work. Any student suspected of plagiarism will be subject to the procedures set out by the Faculty/University (including failing the course entirely).
    - **Academic Integrity:** The University’s policies on academic integrity will be enforced against students who violate the University's standards of academic integrity. Examples of behaviour that are not allowed are:
        - Copying all or part of someone else's work and submitting it as your own;
        - Giving another student in the class a copy of your work and
        - Copying parts from the internet, textbooks, etc.
        - If you have any questions concerning what is allowed, please don’t hesitate to discuss them with me.
    - **We understand that you might positively refer to AI tools to support your research. Please note that, in case there are any concerns about AI-generated submission content, you will be invited for an opportunity to verify and defend your work.**

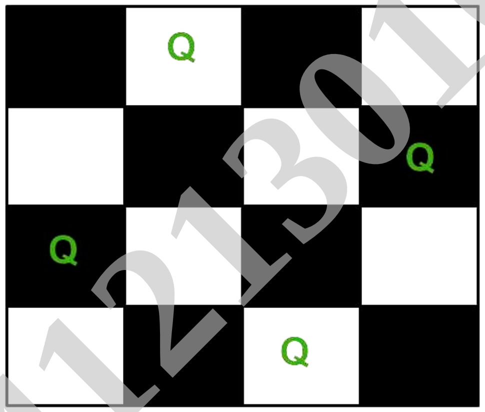
# AI310/CS361 ARTIFICIAL INTELLIGENCE FALL 2025

## COURSE PROJECT DESCRIPTIONS (17 IDEAS)

## Table of Contents

## 1) N-Queens Problem Solver (for different sizes – n should be selected by the user) using the Backtracking Search Algorithm, a Best-First Search, a Hill-Climbing Search, AND a Cultural Algorithm.

**N-Queens Problem Solver Project Overview:** The N-Queens Problem Solver project aims to develop an intelligent system capable of solving the N-Queens problem for various board sizes. The N-Queens problem is a classic chessboard puzzle in which the objective is to place N queens on an N×N chessboard in such a way that no two queens threaten each other. Threatening means no two queens share the same row, column, or diagonal. The value of N, representing the board size and the number of queens, can be chosen by the user. For example, the following is a solution for the 4 Queens problem:



**Problem Description:** In the N-Queens problem, the challenge is to arrange N queens on an N×N chessboard, ensuring that no two queens attack each other. The difficulty arises from the restrictive nature of queen movements: they can traverse horizontally, vertically, or diagonally. The project involves exploring multiple algorithms to efficiently find solutions to this puzzle.

**Backtracking Search Algorithm:** The Backtracking Search Algorithm is a systematic method for exploring potential solutions to a problem. In the context of the N-Queens problem, it involves placing queens on the board one by one and backtracking if a conflict is detected. This algorithm guarantees finding all possible solutions.

**Best-First Search Algorithm:** Best-First Search is an algorithm that intelligently selects the most promising path based on a heuristic evaluation. In the N-Queens context, this algorithm evaluates board configurations using a heuristic to guide the placement of queens, prioritising paths that seem most likely to lead to a solution.

**Hill-Climbing Search Algorithm:** Hill-Climbing is a local search algorithm that continually moves towards higher elevations in the solution space. Applied to the N-Queens problem, it involves iteratively adjusting queen placements to ascend towards a configuration with fewer conflicts. It may get stuck in local optima.

**Cultural Algorithm:** The Cultural Algorithm (CA) is an evolutionary computation technique (similar to the Genetic Algorithm) that mimics the dual inheritance found in human societies—where both genetic (population-level) evolution and cultural (knowledge-based) evolution occur in tandem. In CA, two primary spaces are maintained:

- Population Space: This is similar to conventional evolutionary algorithms, where a set of candidate solutions (individuals) evolves using operators like selection, crossover, and mutation.

- Belief Space: This space stores accumulated knowledge (or “culture”) from previous generations. It guides the search process by influencing how new candidates are generated.

**User Interaction:** The user has the flexibility to choose the size of the chessboard (N), making the project adaptable to different scenarios. The system will present solutions generated by each algorithm, allowing the user to compare their effectiveness and efficiency. Additionally, the project provides insights into the strengths and limitations of diverse search and optimisation techniques.

This project not only addresses the fundamental challenge of the N-Queens problem but also serves as an educational tool for understanding and comparing various search algorithms in artificial intelligence. It combines classic problem-solving techniques with contemporary optimisation strategies.

## 2) A Sudoku Puzzle Solver using both the Backtracking Algorithm AND the Cultural Algorithm.

**Description:** The project aims to create an intelligent Sudoku puzzle solver utilising two distinct algorithms: the Backtracking Algorithm and the Cultural Algorithm. Sudoku is a popular number-placement puzzle in which a 9x9 grid must be filled with digits from 1 to 9, ensuring that each row, column, and 3x3 subgrid contains all digits without repetition. Solving Sudoku involves exploring possible configurations and finding a combination that satisfies all constraints. For example, the (left) figure demonstrates a typical Sudoku puzzle, and its solution (right).

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 5 | 3 | | | 7 | | | | |
| 6 | | | 1 | 9 | 5 | | | |
| | 9 | 8 | | | | | 6 | |
| 8 | | | | 6 | | | | 3 |
| 4 | | | 8 | | 3 | | | 1 |
| 7 | | | | 2 | | | | 6 |
| | 6 | | | | | 2 | 8 | |
| | | | 4 | 1 | 9 | | | 5 |
| | | | | 8 | | | 7 | 9 |

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 5 | 3 | 4 | 6 | 7 | 8 | 9 | 1 | 2 |
| 6 | 7 | 2 | 1 | 9 | 5 | 3 | 4 | 8 |
| 1 | 9 | 8 | 3 | 4 | 2 | 5 | 6 | 7 |
| 8 | 5 | 9 | 7 | 6 | 1 | 4 | 2 | 3 |
| 4 | 2 | 6 | 8 | 5 | 3 | 7 | 9 | 1 |
| 7 | 1 | 3 | 9 | 2 | 4 | 8 | 5 | 6 |
| 9 | 6 | 1 | 5 | 3 | 7 | 2 | 8 | 4 |
| 2 | 8 | 7 | 4 | 1 | 9 | 6 | 3 | 5 |
| 3 | 4 | 5 | 2 | 8 | 6 | 1 | 7 | 9 |

**Backtracking Algorithm:** The Backtracking Algorithm is a systematic approach to problem-solving, particularly effective for constraint-satisfaction problems such as Sudoku. It explores possible solutions incrementally, placing digits in empty cells and backtracking when conflicts arise. In the context of Sudoku, the Backtracking Algorithm will intelligently fill cells, ensuring they adhere to the puzzle's rules.

**Cultural Algorithm:** The Cultural Algorithm (CA) is an evolutionary computation technique (similar to the Genetic Algorithm) that mimics the dual inheritance found in human societies, where both genetic (population-level) evolution and cultural (knowledge-based) evolution occur in tandem. In CA, two primary spaces are maintained:

- Population Space: This is similar to conventional evolutionary algorithms, where a set of candidate solutions (individuals) evolves using operators like selection, crossover, and mutation.

- Belief Space: This space stores accumulated knowledge (or “culture”) from previous generations. It guides the search process by influencing how new candidates are generated.

**Project Requirements:**

**User Interaction:** The project will provide a user-friendly interface where individuals can input a Sudoku puzzle or select from pre-existing ones.

**Algorithm Integration:** Both the Backtracking Algorithm and the Cultural Algorithm will be implemented and integrated into the solver. Users can choose which algorithm to employ.

**Solution Visualisation:** The solver will visually display the step-by-step process of solving the Sudoku puzzle, allowing users to understand the algorithm's decision-making.

**Customisation:** Users can customise the size of the Sudoku grid (e.g., 4x4, 6x6, 9x9) to cater to various puzzle complexities.

**Performance Metrics:** The solver will provide performance metrics, such as the number of iterations, to help users evaluate the efficiency of each algorithm.

This project not only enhances understanding of fundamental algorithms but also provides a practical tool for Sudoku enthusiasts to effortlessly solve puzzles of varying complexities. Users can experiment with different algorithms, gaining insights into their strengths and limitations in tackling Sudoku challenges.

## 3) Knight's Tour Problem Solver (for different sizes – n should be selected by the user) using the Backtracking Search Algorithm, AND the Cultural Algorithm.

The Knight's Tour is a captivating chess problem where the challenge is to find a sequence of moves for a knight to visit every square on the chessboard exactly once. This project, "Knight's Tour Problem Solver," aims to develop a software solution that efficiently solves the intriguing Knight's Tour problem using two distinct approaches: the Backtracking Search Algorithm and a Cultural Algorithm.

**Problem Overview:** In the Knight's Tour, the knight, a chess piece capable of unique L-shaped moves, must traverse the chessboard, covering each square precisely once. The problem is both combinatorial and algorithmic, requiring strategic exploration to determine a viable sequence of knight moves.

**Approaches:**

**(a) Backtracking Search Algorithm:** Backtracking involves systematically exploring potential moves while intelligently abandoning paths that lead to dead ends. This algorithm efficiently navigates the knight through the chessboard. The Backtracking Search Algorithm is a systematic and resource-efficient method for solving the Knight's Tour. It optimises exploration by discarding unfruitful paths, ensuring a feasible solution.

**(b) Cultural Algorithm:** The Cultural Algorithm (CA) is an evolutionary computation technique (similar to the Genetic Algorithm) that mimics the dual inheritance found in human societies, where both genetic (population-level) evolution and cultural (knowledge-based) evolution occur in tandem. In CA, two primary spaces are maintained:

- Population Space: This is similar to conventional evolutionary algorithms, where a set of candidate solutions (individuals) evolves using operators like selection, crossover, and mutation.

- Belief Space: This space stores accumulated knowledge (or "culture") from previous generations. It guides the search process by influencing how new candidates are generated.

The algorithm adapts over successive generations to find an optimal knight's tour. Cultural Algorithms adopt a stochastic, adaptive approach to solving the Knight's Tour. By exploring diverse move sequences and evolving over generations, the algorithm aims to discover effective paths.

**Requirements:**

**User-Selected Board Size:** An additional feature of this project is user customisation. Users can select the size of the chessboard (n), allowing them to explore the Knight's Tour for different board dimensions. This flexibility enhances the project's applicability and provides users with a tailored experience.

**Interactive Interface:** Develop a user-friendly interface that allows users to enter the chessboard size (n) and visualise the knight's tour solutions.

**Backtracking Implementation:** Implement the Backtracking Search Algorithm to systematically explore and find a solution to the Knight's Tour for the user-defined board size.

**Cultural Algorithm Implementation:** Design and integrate a Cultural Algorithm to provide an alternative solution strategy that fosters diversity and adaptability.

**Visualisation:** Include a graphical representation of the knight's movements on the chessboard, aiding users in understanding and appreciating the solutions generated.

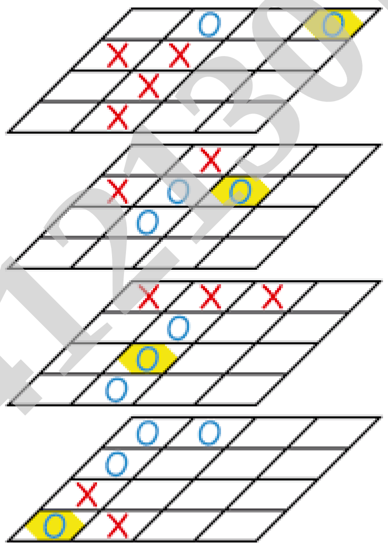
**Optimisation:** Strive for optimised algorithms to handle larger chessboard sizes efficiently.

This project aims not only to solve the Knight's Tour problem but also to offer users an engaging, interactive experience, allowing them to explore the fascinating world of chessboard traversal using advanced algorithms.

## 4) An Intelligent Cubic Player using the Minimax Algorithm, Alpha-Beta Pruning, and Heuristic Functions.

**Introduction:** The project aims to create an intelligent player for the Cubic game, a variant of Tic-Tac-Toe played on a 4x4x4 three-dimensional grid. The player will be designed to make strategic moves using artificial intelligence techniques, specifically implementing the Minimax algorithm, Alpha-Beta Pruning, and heuristic functions.

**Problem Description:** Cubic is an engaging board game that introduces an additional layer of complexity compared to traditional Tic-Tac-Toe. Played on a 4x4x4 grid, each player takes turns placing their markers in an attempt to create a line of four in any direction within the three dimensions. The challenge lies in the spatial nature of the game, requiring players to think strategically in three dimensions to secure a winning position.



**Project Requirements:**

**Minimax Algorithm Implementation:** Develop the Minimax algorithm tailored for the Cubic game. The algorithm should traverse the three-dimensional game space, evaluating possible moves and selecting the optimal move that maximises the chances of winning or minimises the chances of losing.

**Alpha-Beta Pruning Integration:** Implement Alpha-Beta Pruning alongside the Minimax algorithm to enhance computational efficiency. Alpha-Beta Pruning eliminates unnecessary evaluations in the game tree, reducing the time needed to find the best move.

**Heuristic Functions Design:** Design heuristic functions specific to Cubic to provide a quick assessment of the current board state. Heuristics should capture spatial patterns and configurations that lead to winning positions, aiding the AI player in decision-making.

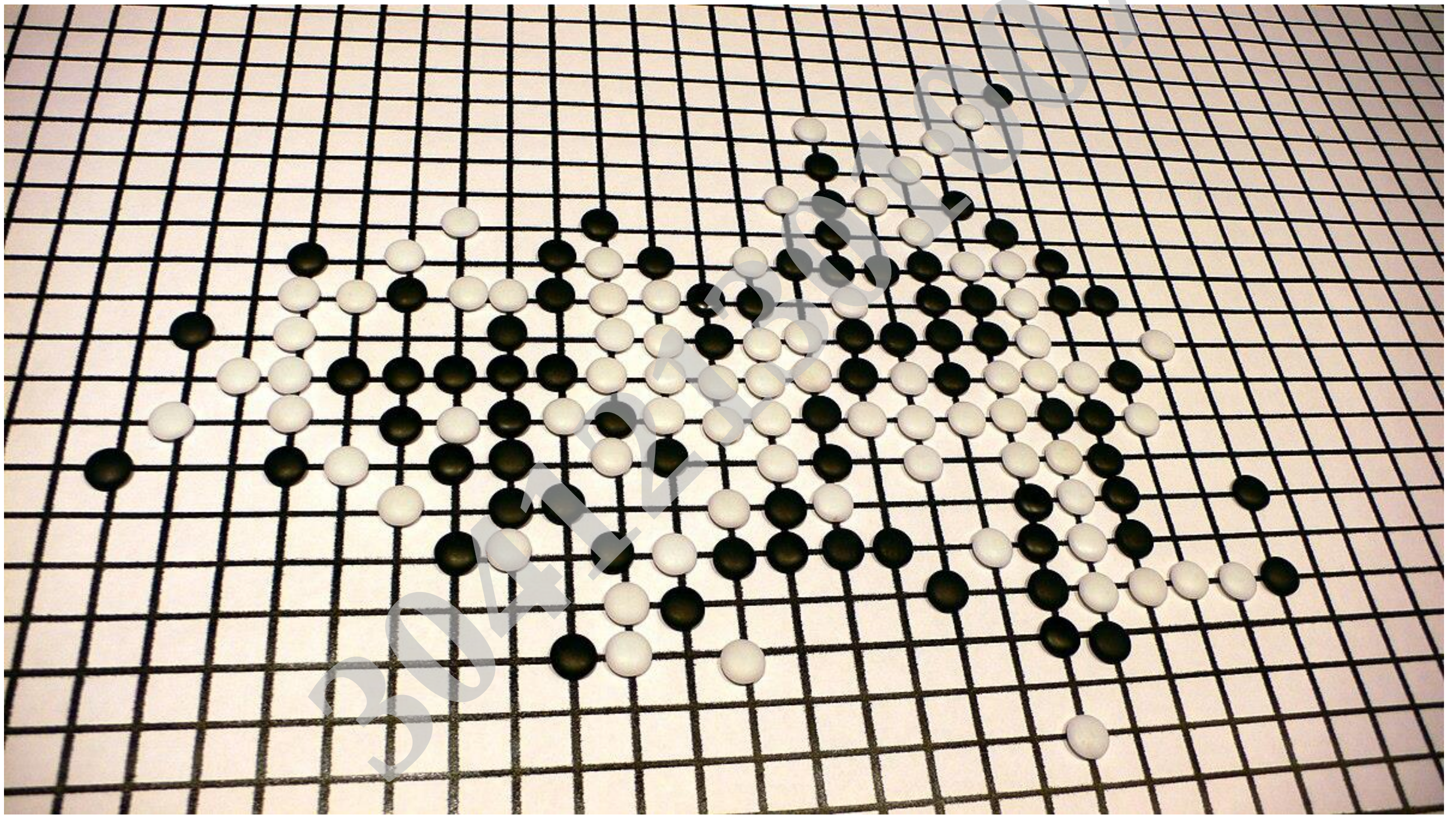
**User Interface Development:** Create a user-friendly interface for interacting with the Cubic player. The interface should allow users to make moves on the 4x4x4 grid and display the AI's decisions clearly (i.e., in an understandable manner), enhancing the overall user experience.

**Optimisation:** Optimise algorithms for responsiveness and efficiency, considering the computational demands of the three-dimensional game space.

**Expected Outcome:** The project's successful outcome will be an intelligent Cubic player capable of making strategic moves on the 4x4x4 grid using the implemented Minimax algorithm, Alpha-Beta Pruning, and heuristic functions. The player should enjoy a challenging, engaging gaming experience that showcases artificial intelligence's ability to master three-dimensional spatial reasoning.

## 5) An Intelligent Connect-6 Player using the Minimax Algorithm, Alpha-Beta Pruning, and Heuristic Functions.

**Description:** The project aims to develop an intelligent player for the Connect-6 game, an extended version of Connect Four, where players strive to connect six markers in a row horizontally, vertically, or diagonally on a 19x19 grid. This board game introduces increased complexity compared to Connect Four, making it a challenging problem for artificial intelligence (AI) systems to navigate strategically.



**Project Requirements:**

**Minimax Algorithm Implementation:** The project requires implementing the Minimax algorithm, a decision-making approach commonly used in two-player games. Minimax explores potential moves by recursively evaluating game states to determine the optimal strategy for maximising wins or minimising losses.

**Alpha-Beta Pruning Integration:** Integrating Alpha-Beta Pruning with the Minimax algorithm is essential for optimising the AI player's decision-making process. Alpha-Beta Pruning helps reduce the number of evaluated game states, making the algorithm more efficient without compromising strategic depth.

**Heuristic Function Design:** Develop heuristic functions that provide a quick evaluation of a given board state. Heuristics should capture patterns and configurations conducive to winning, guiding the AI player's decisions when exhaustive search is impractical.

**User-Selected Board Size (n):** Allow users to select the size of the Connect-6 board (n), introducing flexibility and adaptability to different game scenarios. The AI player should be able to adjust its strategy based on the chosen board size, accommodating variations in complexity.

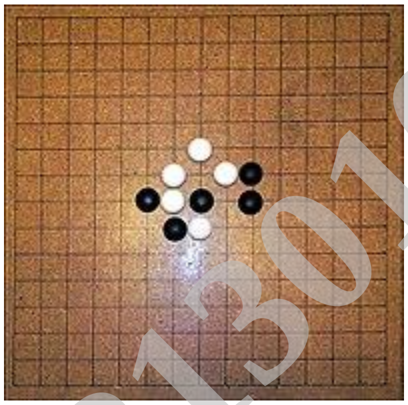
**User-Friendly Interface:** Design a user-friendly interface that allows players to interact seamlessly with the intelligent Connect-6 player. The interface should facilitate user input on the desired board size and enable them to play a strategic game against the AI.

**Optimisation and Performance:** Optimise the AI algorithms for performance, considering computational efficiency and responsiveness. Strive to create an intelligent player that balances strategic depth with real-time responsiveness during gameplay.

**Conclusion:** This project aims to deliver an intelligent Connect-6 player that combines the strategic depth of Minimax, the efficiency of Alpha-Beta Pruning, and the adaptability of heuristic functions. The implementation should provide users with a challenging, engaging gaming experience that showcases AI's capabilities in strategic decision-making within the context of board games.

## 6) An Intelligent Gomoku Player using the Minimax Algorithm, Alpha-Beta Pruning, and Heuristic Functions.

**Description:** The goal of this group project is to create a smart player for the Gomoku game using advanced AI techniques. Gomoku, also known as Five in a Row, is a classic board game where two players take turns placing their markers on a (usually 15x15) grid. The objective is to be the first to achieve a continuous sequence of five markers either horizontally, vertically, or diagonally.



**Game Rules:**

***Board***: *Gomoku is typically played on a 15×15 grid, though variations with different board sizes exist.*

***Players***: *The game is played between two players; Black plays first.*

***Turns***: *Players take turns placing their symbols on vacant intersections. Players take turns placing their markers (usually black and white stones or "X" and "O") on the empty grid.*

***Objective***: *The first player to achieve an unbroken row of five symbols wins. The row can be horizontal, vertical, or diagonal.*

***Blocked Board***: *If the board is filled without a winner, the game is a draw.*

*In some rules, this line must be exactly five stones long; six or more stones in a row does not count as a win and is called an overline.*

**Project Requirements:**

**Minimax Algorithm Implementation:** Develop the Minimax algorithm, a strategic decision-making tool that examines potential moves and predicts outcomes. It ensures the AI player makes moves that maximise its chances of winning and minimise its chances of losing.

**Alpha-Beta Pruning Integration:** Integrate Alpha-Beta Pruning with Minimax for efficient decision-making. Alpha-Beta Pruning helps the AI player disregard unpromising branches in the decision tree, optimising the search process.

**Heuristic Function Design:** Create heuristic functions that quickly evaluate the current state of the game. These functions should capture strategic elements like potential threats, defensive moves, and advantageous positions to guide the AI's decision-making.

**User Interface:** Design a user-friendly interface that allows players to interact with the intelligent Gomoku player. The interface should display the game board, accept player moves, and provide a seamless experience.

**Adaptive Difficulty Levels:** Implement varying difficulty levels that adapt to different player skill levels. The AI player should dynamically adjust its strategy, offering a challenging experience for both beginners and experienced players.

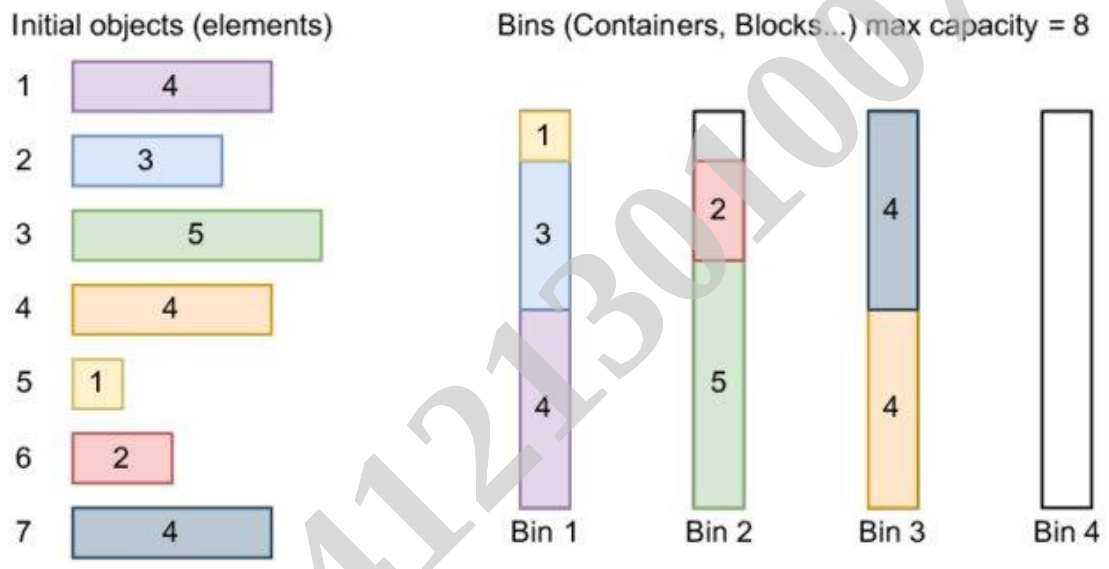
**User-Selected Board Size (n):** Allow users to choose the size of the Gomoku board (n), providing flexibility for different gaming scenarios. The AI player should be able to adapt to various board sizes.

**Optimisation:** Optimise the AI algorithms to balance strategic sophistication and responsive performance. The intelligent Gomoku player should provide real-time responses during gameplay.

**Conclusion:** The Intelligent Gomoku Player project aims to deliver a sophisticated AI opponent for the Gomoku game. By implementing Minimax, Alpha-Beta Pruning, and heuristic functions, the project aims to provide users with an enjoyable and challenging gaming experience while showcasing AI's capabilities in strategic board games.

## 7) Bin Packing Problem Solver using the Backtracking Search Algorithm, AND a Cultural Algorithm.

**Description:** The Bin Packing Problem is a classic optimisation problem in which the objective is to efficiently pack a set of items into a minimum number of bins or containers. Each item has a specific size, and the bins have a limited capacity. The goal is to find an arrangement that minimises the number of bins used. This problem has real-world applications in logistics, resource allocation, and space optimisation.


Initial objects (elements)
Bins (Containers, Blocks...) max capacity = 8
1
4
2
3
3
5
4
4
5
1
6
2
7
4
1
3
4
2
5
4
4
Bin 1
Bin 2
Bin 3
Bin 4

**Problem Rules:**

*__Items__: There is a set of items, each with a defined size or volume.*

*__Bins__: A limited number of bins or containers are available, each with a maximum capacity.*

*__Packing Objective__: The objective is to pack all items into the bins in a way that minimises the number of bins used.*

*__Item Placement__: Items cannot be split or partially placed in different bins; they must be placed whole.*

*__Bin Capacity__: The total size of items in a bin cannot exceed its maximum capacity.*

**Project Requirements**: The project aims to develop a Bin Packing Problem Solver using two different approaches: the Backtracking Search Algorithm and a Cultural Algorithm. The solver will address the following key components:

**Backtracking Search Algorithm:** Develop the Backtracking Algorithm to explore possible combinations of item placements in bins. Enhance the algorithm to prune branches and efficiently explore feasible solutions.

**Cultural Algorithm:** The Cultural Algorithm (CA) is an evolutionary computation technique (similar to the Genetic Algorithm) that mimics the dual inheritance found in human societies, where both genetic (population-level) evolution and cultural (knowledge-based) evolution occur in tandem. In CA, two primary spaces are maintained:

- Population Space: This is similar to conventional evolutionary algorithms, where a set of candidate solutions (individuals) evolves using operators like selection, crossover, and mutation.

- Belief Space: This space stores accumulated knowledge (or "culture") from previous generations. It guides the search process by influencing how new candidates are generated.

Design a CA representation of bin configurations and explore genetic operators like mutation and crossover. Implement a fitness function to evaluate the quality of a given bin configuration. Allow the CA to evolve populations toward better solutions.

**User Interface:** Enable users to input item sizes, bin capacities, and other relevant parameters.

**Algorithm Selection:** Provide users with the option to choose between the Backtracking Algorithm and the Cultural Algorithm.

**Visual Representation:** Implement a visual representation of the bin-packing solutions generated by both algorithms.

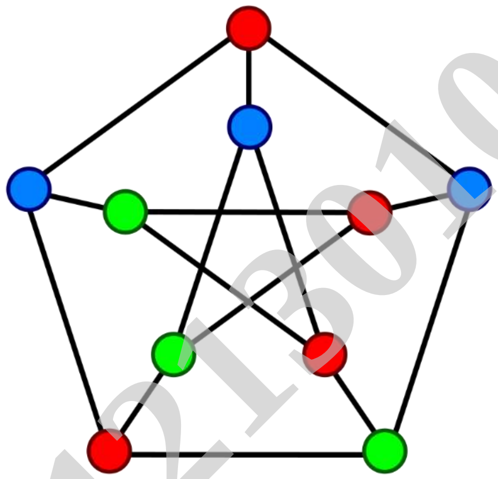
**Comparison Display:** Allow users to compare the solutions produced by the two algorithms.

**Performance Metrics:** Implement metrics to assess the efficiency and effectiveness of both algorithms. Analyse and display comparative results, such as solution quality and computation time.

**Outcome:** The project's success will be measured by the ability of the developed solver to efficiently and accurately solve the Bin Packing Problem using both the Backtracking Search Algorithm and the Cultural Algorithm. The user interface should provide a clear understanding of the solutions generated by each algorithm, enabling users to compare their performance.

## 8) Graph Colouring Problem Solver using the Backtracking Search Algorithm, AND the Cultural Algorithm.

**Introduction:** The Graph Colouring Problem is a well-known combinatorial optimisation challenge that involves assigning colours to the vertices of a graph in such a way that no two adjacent vertices share the same colour. This project aims to develop an intelligent Graph Colouring Problem Solver utilising both the Backtracking Search Algorithm and a Cultural Algorithm.



**Problem Description:** In the Graph Colouring Problem, we have a graph composed of vertices and edges. The objective is to colour each vertex in a way that no two connected vertices have the same colour. The minimum number of colours required to achieve this is known as the chromatic number of the graph. Our task is to find a valid colouring solution that minimises the chromatic number.

**Backtracking Search Algorithm**: The Backtracking Search Algorithm is a systematic exploration approach that examines different colour assignments for each vertex. It works by recursively trying out colours and backtracking when conflicts arise. This algorithm explores the solution space while intelligently pruning paths that lead to infeasible colourings.

**Cultural Algorithm:** The Cultural Algorithm (CA) is an evolutionary computation technique (similar to the Genetic Algorithm) that mimics the dual inheritance found in human societies, where both genetic (population-level) evolution and cultural (knowledge-based) evolution occur in tandem. In CA, two primary spaces are maintained:

- Population Space: This is similar to conventional evolutionary algorithms, where a set of candidate solutions (individuals) evolves using operators like selection, crossover, and mutation.

- Belief Space: This space stores accumulated knowledge (or "culture") from previous generations. It guides the search process by influencing how new candidates are generated.

In this context, we represent potential colourings as chromosomes. The algorithm evolves a population of colourings over generations using genetic operations such as crossover and mutation. The fittest colourings, those with fewer conflicts, are more likely to be passed to the next generation.

**Project Requirements:**

**User-Defined Graphs:** The system should allow users to input graphs with specified vertices and edges.

**Backtracking Implementation:** Develop a Backtracking Search Algorithm to find valid colourings for the given graph.

**Cultural Algorithm Implementation:** Implement a Cultural Algorithm that evolves colourings over generations.

**User Interface:** Create an intuitive interface that allows users to input graphs, select algorithms, and visualise colourings.

**Solution Visualisation:** Provide a graphical representation of the graph with coloured vertices to demonstrate the algorithm's effectiveness.

**Performance Metrics:** Implement metrics to measure the quality of solutions, such as the chromatic number and computational time.

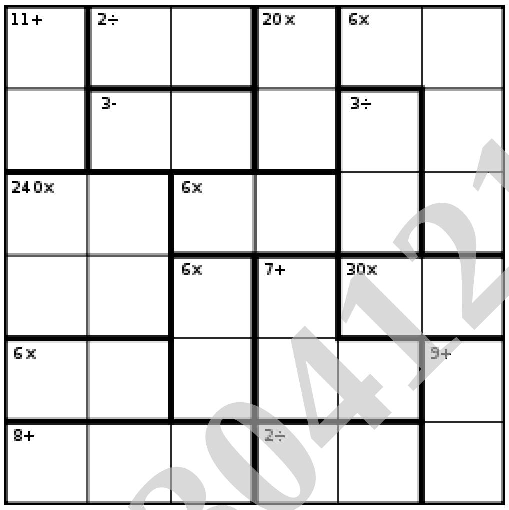
**Parameter Tuning:** Allow users to adjust parameters of the cultural algorithm, such as population size, mutation rates, and belief space size, to explore their impact on solution quality.

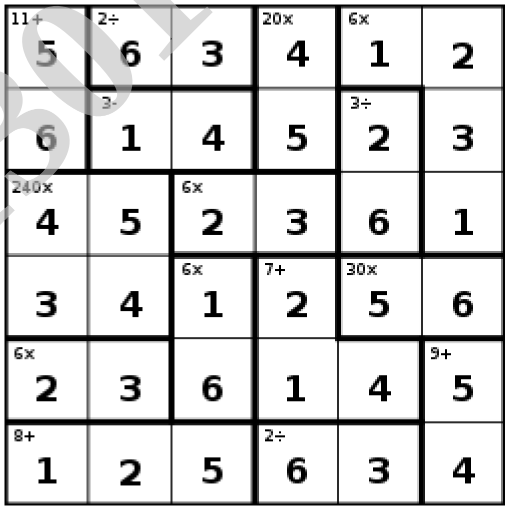
**Outcome:** The project aims to deliver a versatile and user-friendly Graph Colouring Problem Solver that demonstrates the strengths of both the Backtracking Search Algorithm and the Cultural Algorithm. Users will gain insights into how different algorithms approach and optimise the colouring of graphs, contributing to a deeper understanding of combinatorial optimisation challenges.

## 9) KenKen (KenDoku) Puzzle Solver using the Backtracking Search Algorithm, AND a Cultural Algorithm.

The KenKen Puzzle is a mathematical and logical grid-based puzzle that combines aspects of Sudoku and arithmetic operations. The goal is to fill a grid with digits so that each row and column contains unique numbers, and certain outlined regions (cages) follow specified arithmetic rules.

**Problem/Game Description:** In a KenKen Puzzle, you are presented with an N×N grid, where N is usually a perfect square (e.g., 4, 6, 9). The grid is divided into regions or cages, each outlined with bold borders. Each cage contains a target number and an arithmetic operation (addition, subtraction, multiplication, or division).





**Rules of the KenKen Puzzle:**

**Digit Placement:** Fill the grid with digits from 1 to N in such a way that each row and each column contains unique digits.

**Cage Operations:** The digits within each cage must satisfy the specified arithmetic operation to achieve the target number. For example, a cage labelled "8÷" with a target of 2 means that the digits in the cage must divide to give 2. Thus, the numbers in the cells of each cage must produce a particular “target” number when combined using a specified mathematical operation (one of addition, subtraction, multiplication or division). Another example, a linear three-cell cage specifying addition and a target number of 6 in a 4×4 puzzle must be satisfied with the digits 1, 2, and 3. Digits may be repeated within a cage, as long as they are not in the same row or column. The target number and operation appear in the upper left-hand corner of the cage.

**Requirements of the Project:** The objective of the KenKen Puzzle Solver Project is to design an intelligent system capable of solving KenKen puzzles using both the Backtracking Search Algorithm and a Cultural Algorithm.

**Project Components:**

**Backtracking Search Algorithm:** Implement the Backtracking Algorithm to explore possible digit placements in the grid. The algorithm should backtrack when constraints are violated and efficiently navigate through the solution space.

**Cultural Algorithm (CA):** Develop a Cultural Algorithm to approach KenKen puzzle-solving as an evolutionary process. Represent potential solutions as individuals in a population, apply genetic operators, and evolve the population over generations to find optimal or near-optimal solutions.

**Integration:** Integrate the Backtracking Algorithm and the Cultural Algorithm into a cohesive system. Allow users to choose between the two methods for puzzle-solving.

**User Interaction:** Create a user-friendly interface that allows users to input KenKen puzzles, visualise solutions, and observe the solving process.

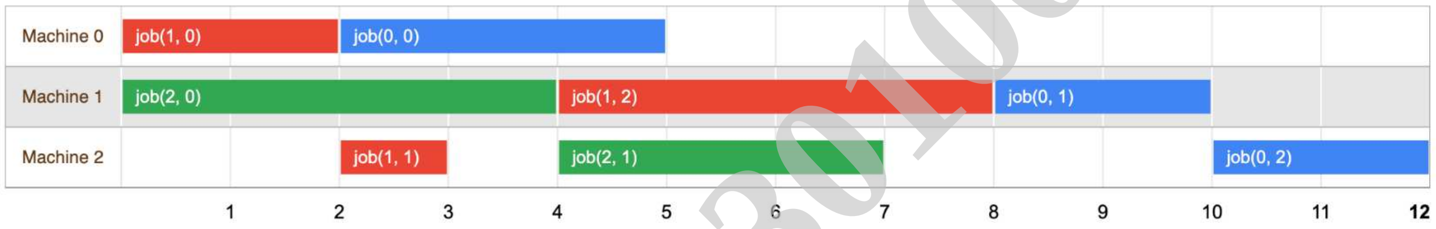
**Performance Metrics:** Implement metrics to evaluate the efficiency and accuracy of the solver, considering factors such as solution time and correctness.

**Challenges:** Consider the challenges associated with implementing Cultural Algorithms for KenKen puzzles, including suitable representations, genetic operators, and the handling of constraints.

**Outcome:** Upon completion, the KenKen Puzzle Solver Project will serve as a valuable tool for enthusiasts and puzzle-solvers, showcasing the application of both Backtracking and Cultural Algorithms in tackling mathematical and logical challenges. Users can gain insights into the strengths and weaknesses of each algorithmic approach and witness their performance in real puzzle-solving scenarios.

## 10) Job Scheduling Problem Solver using the Backtracking Search Algorithm, AND a Cultural Algorithm.

**Description**: The Job Scheduling Problem (JSP) is a classic optimisation challenge encountered in various industries, where the goal is to efficiently assign a set of jobs to available resources while respecting constraints and optimising a specific objective. The project aims to develop intelligent algorithms, specifically employing the Backtracking Search Algorithm and a Cultural Algorithm, to solve instances of the Job Scheduling Problem.



**Problem Overview**: In the Job Scheduling Problem, a set of jobs must be scheduled onto available resources, subject to constraints such as time, resource capacity, and dependencies between jobs. Each job has specific processing times and may require certain resources, and the objective is to find an optimal schedule that minimises the completion time or maximises resource utilisation.

**Backtracking Search Algorithm**: The Backtracking Search Algorithm is a systematic approach that explores different job assignments recursively. It starts with an initial assignment, explores possibilities, and backtracks when constraints are violated. The algorithm incrementally builds a schedule, making decisions at each step, and aims to find a valid and optimal solution.

**Cultural Algorithm**: The Cultural Algorithm (CA) is an evolutionary computation technique (similar to the Genetic Algorithm) that mimics the dual inheritance found in human societies, where both genetic (population-level) evolution and cultural (knowledge-based) evolution occur in tandem. In CA, two primary spaces are maintained:

- Population Space: This is similar to conventional evolutionary algorithms, where a set of candidate solutions (individuals) evolves using operators like selection, crossover, and mutation.

- Belief Space: This space stores accumulated knowledge (or "culture") from previous generations. It guides the search process by influencing how new candidates are generated.

Thus, the CA maintains a population of potential solutions (schedules), applies genetic operators (crossover, mutation), and evaluates the fitness of each schedule based on the optimisation objective. Over successive generations, the algorithm evolves schedules that exhibit desirable traits, leading to improved solutions.

**Project Requirements**:

**Problem Representation**: Define a suitable representation for the Job Scheduling Problem that captures job details, resource constraints, and the objective function.

**Algorithm Implementation**: Implement the Backtracking Search Algorithm to systematically explore the decision space. Additionally, implement a Cultural Algorithm to evolve schedules over multiple generations.

**Constraints Handling**: Develop mechanisms to handle constraints, including resource capacity, temporal constraints, and any specific rules governing job assignments.

**Optimisation Objective**: Clearly define the optimisation objective, whether it's minimising the makespan, maximising resource utilisation, or another relevant metric.

**Performance Evaluation**: Evaluate the performance of both algorithms on various instances of the Job Scheduling Problem, comparing their effectiveness in finding optimal schedules and their computational efficiency.

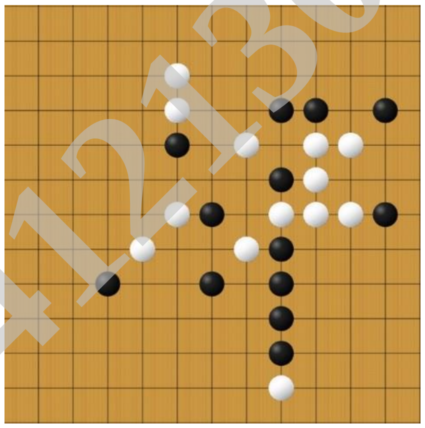
**User Interface**: If feasible, consider implementing a user-friendly interface allowing users to input problem instances, visualise schedules, and understand algorithmic decisions.

**Expected Outcomes**: The project aims to deliver effective solutions to the Job Scheduling Problem using both Backtracking and Cultural Algorithms. The team is expected to provide insights into the strengths and limitations of each algorithm, helping users make informed decisions when selecting approaches for specific problem instances. The project will contribute to understanding and efficiently solving real-world scheduling challenges.

## 11) An Intelligent Pente Player using the Minimax Algorithm, Alpha-Beta Pruning, and Heuristic Functions.

**Introduction:** The project aims to develop an intelligent Pente player; a classic two-player strategy board game. Pente, derived from the Japanese word for "five," involves placing stones on a gridded board with the goal of getting five stones in a row. The game is known for its strategic depth, requiring players to plan ahead and block opponents' moves. The intelligent player will be implemented using the Minimax algorithm, enhanced with Alpha-Beta Pruning for efficiency, and heuristic functions for quicker decision-making.

**Game Description:** Pente is played on a gridded Go board, typically 19x19 intersections. Players take turns placing stones of their colour on the intersections (White always assumes the opening move), aiming to form an unbroken row of five stones horizontally, vertically, or diagonally. Players can also capture their opponent's stones by surrounding them with their own stones. The first player to achieve five in a row or capture ten opponent stones wins.



**Project Requirements:**

**Game Representation:** Define a data structure to represent the Pente board and track the positions of stones for both players.

**Move Generation:** Implement algorithms to generate legal moves for both players, considering the rules of stone placement and capture.

**Minimax Algorithm:** Integrate the Minimax algorithm, a decision-making approach that explores possible moves and evaluates outcomes to make optimal decisions.

**Alpha-Beta Pruning:** Enhance the Minimax algorithm with Alpha-Beta Pruning to optimise the search and reduce the number of explored nodes.

**Heuristic Evaluation Functions:** Design heuristic functions to evaluate the desirability of a game state, capturing strategic elements and patterns that influence decision-making.

**User Interface:** Create a user interface to visualise the Pente board, allowing users to play against the intelligent Pente player and observe its moves.

**Expected Outcomes:** The project aims to deliver an intelligent Pente player capable of strategic decision-making, blocking opponents effectively, and demonstrating the strengths of the Minimax algorithm with Alpha-Beta Pruning and heuristic functions. The team will evaluate the intelligent player’s performance through test games against human players and, if possible, other AI opponents. Success will be measured by the player's ability to make optimal moves, anticipate opponent strategies, and adapt to different game scenarios. The user interface, if developed, will provide an interactive platform for users to engage with the intelligent Pente player.

## 12) Sports Tournament Scheduling via Genetic Algorithms (GA).

- Represent tournament schedules (e.g., round-robin or knockout formats) as individuals in the GA population.
- Define objectives to minimise venue conflicts, ensure fair rest periods, and balance game times.
- Develop fitness functions to score schedules based on constraint violations.
- Implement genetic operators (crossover, mutation) tailored to scheduling order.
- Validate the approach by comparing generated schedules with a baseline solution.

**Detailed Description:**

**Context and Problem Statement:** Organising sports tournaments involves managing multiple constraints, such as venue availability, rest periods, and game order. Using GA, teams can evolve schedules that satisfy these constraints optimally.

**Key Terms and Concepts:**

- *Tournament Scheduling:* The process of planning matches such that all teams play under fair conditions.
- *Genetic Algorithm (GA):* An optimisation method inspired by the process of natural selection.
- *Fitness Evaluation:* A way to quantify how well a given schedule meets requirements.

**Requirements/Deliverables:**

- A prototype Python tool that generates and refines tournament schedules using GA.
- Simulation results comparing different GA parameter settings.
- Visual scheduling charts (e.g., calendars, tables) to illustrate the final schedule.
- A report discussing challenges, solutions, and possible real-world applications.

## 13) Solving a Faculty's Timetable Scheduling Problem using the Cultural Algorithm.

A very famous scenario where Evolutionary Algorithms can be used is the process of making timetables or timetable scheduling. Consider you are trying to come up with a weekly timetable for classes in a college for a batch/class. We must arrange classes and come up with a timetable so that there are no clashes between classes. Here, the task is to search for the optimum timetable schedule. A possible definition for the problem is: Given a set of lecturers, a set of courses on individual topics and a Course Requirements matrix with integer elements representing the number of hours a lecturer teaches a course during each week, the problem is to allocate times to these hours so that a student may take as many suitable combinations of courses as possible. Or, simply to create a practical timetable for a whole faculty in which courses offered by different departments may be combined in various ways to suit individual students.

## 14) Cloud Resource Allocation using the Genetic Algorithm (GA).

- Model a simple cloud environment with a set of tasks and available computing resources.
- Define objectives like minimal cost or response time.
- Implement a cloud resource allocation approach: use GA for candidate solution generation.
- Test the approach on standard workload or simulated data.
- Evaluate the efficiency and robustness of the resource allocation strategy.

**Detailed Description:**

**Context and Problem Statement:** Cloud computing demands efficient resource allocation to balance load and reduce operational costs. This project uses evolutionary algorithms to devise a dynamic resource allocation strategy.

**Key Terms and Concepts:**

- *Resource Allocation:* Assigning computing resources to tasks in an optimal way.
- *Genetic Algorithm (GA):* An algorithm that evolves candidate solutions using techniques inspired by biological evolution.
- *Fitness Evaluation:* Assessing solutions based on cost and performance metrics.

**Requirements/Deliverables:**

- A Python simulation that models cloud tasks and resource constraints.
- An algorithm based on GAs with clearly defined evaluation metrics.
- Experimental data and plots demonstrating how the model outperforms standard models.
- A final report detailing methods, experiments, and potential real-world applications.

## 15) Solving the Knapsack Problem using a Cultural Algorithm *(Solve both the 0-1 Knapsack Problem and the Unbounded Knapsack Problem)*.

The knapsack problem or rucksack problem is a problem in combinatorial optimisation: Given a set of items, each with a weight and a value, determine the number of each item to include in a collection so that the total weight is less than or equal to a given limit and the total value is as large as possible. It derives its name from the problem faced by someone who is constrained by a fixed-size knapsack and must fill it with the most valuable items. The most common problem solved is the 0-1 knapsack problem, which restricts the number of copies of each item type to 0 or 1. The unbounded knapsack problem (UKP) places no upper bound on the number of copies of each kind of item.

## 16) Feature Selection Using Genetic Algorithms to Train Decision Trees.

- Identify a publicly available dataset (e.g., for diagnosing diseases).
- Define an objective function for feature selection that maximises classification accuracy while reducing feature count (i.e., training a decision tree).
- Implement a Genetic Algorithm (GA) where each individual represents a feature subset.
- Run experiments comparing different selection and crossover strategies on the dataset.
- Summarise findings with charts, accuracy metrics, and a final report.

**Detailed Description:**

**Context and Problem Statement:** In machine learning, reducing the number of features can simplify models and improve performance. Selecting the best features is critical for accurate diagnosis.

**Key Terms and Concepts:**

- *Feature Selection:* The process of choosing a subset of relevant features (variables) for model training.
- *Genetic Algorithm (GA):* An algorithm that evolves candidate solutions using techniques inspired by biological evolution.

**Requirements/Deliverables:**

- A GA implementation for selecting a meaningful subset of the dataset features.
- A detailed comparison of different GA operators (selection, crossover, mutation).
- Performance metrics (accuracy, reduction percentage) across multiple runs.
- A comprehensive report with visual aids (graphs and tables) showing algorithm performance.

## 17) Automated OCR of Handwritten English Letters using Feature Engineering and Decision Trees / Random Forests.

Optical character recognition or optical character reader (OCR) is the automated conversion of images of typed, handwritten, or printed text into machine-encoded text, whether from a scanned document, a photo of a document, a scene-photo (e.g., the text on signs and billboards in a landscape photo) or from subtitle text superimposed on an image (e.g., from a television broadcast).

Recognising single handwritten letters is a standard, small-scale supervised classification problem. Decision trees and (especially) Random Forests can work well if you provide good input features (they don’t perform as well directly on high-dimensional raw pixels vs CNNs, but with feature engineering or dimensionality reduction, they give solid baseline results).

**Suggested practical approach:**

- Use a public dataset for letters (e.g., EMNIST letters or a small custom dataset collected in-class).
- Preprocess: resize to a fixed small size (e.g., 28×28), convert to grayscale, and normalise.
- Feature extraction options (pick at least 1–2):
    - HOG features (Histogram of Oriented Gradients) → compact, informative for shapes.
    - PCA to reduce pixel dimensionality (then use principal components as features).
    - Simple handcrafted features: e.g., aspect ratio, stroke-density zones, number of end-points, etc.
- Train a Decision Tree, then a Random Forest and compare to the single Decision Tree baseline.
- Cross-validate, produce a confusion matrix and per-class accuracy.

**Expected outcome:** Working classifier (Decision Tree vs Random Forest) with clear comparison, explanations (feature importance), and error analysis. Reasonable accuracy for single letters (depends on data quality).

You MAY use the following data-set: https://www.kaggle.com/crawford/emnist