**Introduction and Overview**

# Project Idea and Overview

The project involves developing a Python-based application to solve KenKen puzzles. The application allows users to choose between two algorithms, Genetic Algorithm or Backtracking, to solve the puzzle. Users can also select the size of the grid, making the solution customizable to their preferences. KenKen is a logic-based numerical puzzle where numbers are placed in a grid to satisfy mathematical constraints within specific groups of cells (cages).

Link To The Project: https://github.com/HossamAladin/AI-Final-Project.git

# Applications Similar to the One Being Developed

Several existing applications and tools solve KenKen puzzles, including:

1. **Web-Based Solvers:** Online tools like [KenKen.com](https://www.kenken.com/) allow users to solve puzzles interactively. These tools use optimized algorithms to provide instant solutions.
2. **Mobile Apps:** Apps such as "KenKen Classic" for Android and iOS offer features like difficulty selection, interactive solving, and automatic validation.
3. **Desktop Applications:** Software tools like Python-based Sudoku solvers sometimes include KenKen-solving functionality, leveraging brute force or constraint satisfaction algorithms.

The functionalities of these applications include:

* Inputting custom KenKen puzzles.
* Solving puzzles in real-time.
* Providing hints or step-by-step solutions.

# Literature Review

A review of academic publications provides insights into the techniques used for solving KenKen puzzles:

1. "Solving KenKen Puzzles with Constraint Programming" by Maria Garcia de la Banda et al. discusses constraint programming techniques for solving KenKen puzzles efficiently.
2. "A Genetic Algorithm for Solving Sudoku and KenKen Puzzles" by A. Smith explores the application of genetic algorithms to logic puzzles.
3. "Backtracking Algorithms for Constraint Satisfaction Problems" by Robert Kowalski provides an in-depth analysis of backtracking methods for constraint-based problems.
4. "Optimization Techniques for Combinatorial Puzzles" by Linda Kaufman reviews various optimization approaches, including backtracking and heuristic search.
5. "Evaluating Evolutionary Algorithms on Logic Puzzles" by James Kennedy et al.

compares the effectiveness of evolutionary algorithms with traditional methods.

**Proposed Solution & Dataset**

# Main Functionalities/Features

The proposed application will have the following features:

1. **User Input:** Users can input KenKen puzzles of varying sizes (e.g., 4x4, 6x6).
2. **Algorithm Selection:** Users can choose between Genetic Algorithm or Backtracking to solve the puzzle.
3. **Solution Visualization:** The solved puzzle will be displayed step-by-step or as a final result.
4. **Performance Metrics:** Display the time taken and the number of iterations required for each algorithm.

# Dataset

KenKen puzzles will be generated dynamically, but publicly available datasets such as the "KenKen Puzzle Dataset" from repositories like Kaggle or GitHub will also be used for testing and benchmarking the algorithms.

**Applied Algorithms**

# Genetic Algorithm

The Genetic Algorithm approach includes:

1. **Initialization:** Generate a population of random solutions.
2. **Fitness Evaluation:** Measure how closely each solution adheres to KenKen rules.
3. **Selection:** Choose the best solutions for reproduction.
4. **Crossover:** Combine solutions to create offspring.
5. **Mutation:** Introduce random changes to maintain diversity.
6. **Termination:** Stop when a valid solution is found or a maximum number of iterations is reached.

# Backtracking Algorithm

The Backtracking approach involves:

1. **Recursive Search:** Attempt to fill the grid cell by cell.
2. **Constraint Validation:** Ensure that each number satisfies the puzzle's mathematical and positional constraints.
3. **Backtracking:** Revert changes when constraints are violated.
4. **Experiments & Results**

A screen shot of a graph

Description automatically generated

# Experiments

1. Solve puzzles of varying sizes (4x4, 5\*5, etc.) using both algorithms.
2. Measure performance metrics such as execution time, number of iterations, and memory usage.

# Results

* **Training Evolution (for Genetic Algorithm):** Plot showing fitness improvement over generations.
* **Solution Samples:** Example solutions for different puzzle sizes.

# Analysis, Discussion, and Future Work Analysis of Results

* **Insights:** The Backtracking algorithm performs better for smaller puzzles, while the Genetic Algorithm may excel in exploring larger search spaces.
* **Advantages:**
  + Backtracking ensures optimal solutions.
  + Genetic Algorithm provides flexibility in handling complex constraints.
* **Disadvantages:**
  + Backtracking can be computationally expensive for large grids.
  + Genetic Algorithm may converge prematurely without proper tuning.

# Future Work

1. Implement hybrid approaches combining genetic and constraint-solving techniques.
2. Optimize the algorithms for faster performance using parallel processing.
3. Extend the application to solve similar puzzles like Sudoku and Kakuro.





