AI Final Project Report

**Introduction**

The problem we aim to solve is maximizing the score in the game *10x10*. This game is a puzzle-based challenge where the player must place various shapes on a 10x10 grid to complete rows or columns, which are then cleared. The goal is to place as many pieces as possible before the board fills up, making further placements impossible.

This problem is particularly interesting due to its non-intuitive nature. While the rules of the game are simple, the strategy required to maximize the score involves careful planning.  
The problem's complexity makes it a good candidate for applying AI techniques such as search algorithms and heuristics, which we've studied in the course. Additionally, each of us envisioned different approaches to solve this game, making it both an educational and intriguing challenge to implement and compare these strategies.

In this project, we explore several AI techniques to solve the 10x10 game. The strategies we have chosen include:

* **Greedy Agents**: These agents aim to maximize immediate gains by placing pieces in positions that will complete the most rows or columns, without considering the long-term implications.
* **Cautious Agents**: These agents focus on avoiding placements that create small, awkward empty spaces, which would be difficult to fill with future pieces.
* **Pruning Techniques**: We will reduce the search space by eliminating paths that do not lead to optimal solutions, saving computational resources and improving efficiency.
* **Informed Search**: Using heuristic information, we aim to guide the search toward the most promising board configurations, improving search speed and accuracy.

Our initial focus is on developing an optimal way to assign a maximum score to each board state. We will use heuristics to evaluate board states based on criteria such as board emptiness and proximity to completing a line. By drawing inspiration from our earlier coursework, we have identified key ideas for designing these heuristics, such as:

* Assigning higher scores to boards with more empty spaces, as more options are available for future placements.
* Prioritizing boards that are closer to completing rows or columns, thus maximizing the immediate score boost.

Additionally, we plan to experiment with **Reinforcement Learning** techniques, using trial and error to discover optimal strategies through training. Our goal is to develop an agent capable of achieving high scores based on learned patterns from the game.

The challenge is non-trivial due to the random nature of the game and the need to balance immediate gains with long-term viability. By applying search algorithms like Depth-First Search (DFS) and A\* Search, coupled with well-designed heuristics, we aim to solve this problem efficiently.

**Previous Work**

A prior project, *“10x10: AI Final Project Report”* by Guy Hacohen and Amittai Cohen-Zemach (2016), laid a strong foundation for solving the 10x10 puzzle using AI techniques. They approached the problem by designing agents to maximize scores through efficient block placement on the 10x10 board. This project serves as a significant reference for our work, as it outlines strategies and methods relevant to optimizing gameplay performance.

**Game Characteristics and Rules**

In their work, Hacohen and Cohen-Zemach described the core mechanics of the game, emphasizing that the goal is to clear rows and columns by placing blocks in a 10x10 grid. Points are awarded for each block placed and for clearing rows and columns, while the game ends when no further blocks can be placed. This partial observability and randomness make the game challenging and non-trivial to solve optimally.

**Agents and Methods**

Their approach divided the problem into three main parts: defining an order over boards (i.e., scoring each board), pruning irrelevant or poor game states, and searching for optimal placements using heuristic-guided agents. The project implemented various agents, including:

* **Global Search Agents**, such as a Full 3-Move Agent and a Greedy 1-Move Agent, which searched for the best move by evaluating every possible board state.
* **Local Search Agents**, like Hill Climbing Agents, which utilized local optimization techniques to explore neighboring board states.

While the **Full 3-Move Agent** yielded optimal results by evaluating all possible moves, it was computationally expensive. On the other hand, **Greedy Agents** focused on immediate placements, significantly improving runtime at the expense of overall score potential. The **Greedy All-Combinations 1-Move Agent**, which explored different orders of placing blocks, struck a balance between speed and accuracy, outperforming simpler Greedy Agents while maintaining reasonable computational efficiency.

**Pruning and Heuristics**

To improve performance, they employed **pruning techniques** to eliminate unlikely-to-succeed placements early in the search process, thus saving computational resources. Additionally, the project introduced a variety of heuristics to evaluate the “goodness” of a board state. Basic heuristics included counting the number of free spaces, assessing the number of connected components, and evaluating block placements near the center of the board. Advanced heuristics, like evaluating board boundaries and utilizing sub-board analysis, improved the quality of the search results.

To refine their approach, the authors used **genetic algorithms** to optimize the weights of these heuristics, producing a combined heuristic that significantly enhanced the agents' performance.

**Results**

The agents developed in this project achieved high scores, with the **Greedy All-Combinations 1-Move Agent** reaching a score of 89,519, placing it in the top 1% of human players. The project demonstrated that well-designed heuristics and agents can outperform many human players, especially when combined with optimization techniques like genetic search.

This work forms a solid benchmark for our project. We plan to build upon their findings, particularly in the use of *A Search*\* and **DFS**, while further refining heuristic functions to achieve better balance between computation time and score optimization.

Collected Data

Board 10X10

DFS - Avg of 100 games for each score:

|  |  |  |  |
| --- | --- | --- | --- |
| Score | Time (Seconds) | Memory (MB) |  |
| 100 | 0.1791 | 3.5096 |  |
| 1000 | 0.7498 | 11.5741 |  |
| 10000 | 6.7072 | 170.6652 |  |
| 100000 | 170.7096 | 13432.6682 |  |
| 300000 | More then 10 min | More then 45GB | Didn’t finish. Picture below. |

A grid of white squares

Description automatically generated

BFS – 1 game run:

|  |  |  |  |
| --- | --- | --- | --- |
| Score | Time (Seconds) | Memory (MB) |  |
| 10 | 3.03 | 59.646 |  |
| 20 | Not relevant | Not relevant | Too much time and space |

A\* - score heuristic Avg of 100 games for each score:

|  |  |  |  |
| --- | --- | --- | --- |
| Score | Time (Seconds) | Memory (MB) |  |
| 100 | 0.1489 | 5.2174 |  |
| 1000 | 0.8004 | 22.7332 |  |
| 10000 | 19.4957 | 367.7030 |  |
| 100000 |  |  |  |
|  |  |  |  |

A\* - combination of couple heuristics Avg of 100 games for each score:

|  |  |  |  |
| --- | --- | --- | --- |
| Score | Time (Seconds) | Memory (MB) |  |
| 100 |  |  |  |
| 1000 |  |  |  |
| 10000 |  |  |  |
| 100000 |  |  |  |
|  |  |  |  |