Bachelor in Computer Science

COMP 472: Artificial Inteligence

AI WARGAME REPORT

LUCÍA CORDERO SÁNCHEZ

SIMON DUNAND

ALEJANDRO LEONARDO GARCÍA NAVARRO

Instructor: *Leila Kosseim*

Academic Year: 2023/24

Date: December 3rd, 2023

Contents

1 Introduction

This report encapsulates some of the outcomes of our project for COMP472. To begin with, we would like to provide some basic information about the project. An adversarial asymmetric strategy game involving two players (attacker and defender) had to be developed. Each player has a unique set of units, including AI, Viruses, Techs, Programs, and Firewalls, battling on a 5x5 board. The game aims to outmaneuver the opponent by strategically moving and engaging units, ultimately aiming to destroy the opponent's AI unit.

Our task was to implement an adversarial search algorithm to play AI Wargame autonomously. This involved developing and integrating a minimax algorithm with alpha-beta pruning alongside formulating at least two unique heuristics. These were key in guiding the AI's decision-making process during the game, ensuring an efficient and effective strategy. In addition, the program was designed to operate in various modes, including manual, semi-automatic, and fully automatic, with a simple command-line interface for interactions.

In this report, we delve into the details of our heuristics, analyzing their performance and interaction within the scope of the tournament. We explore the impact of alpha-beta pruning on the search's efficiency and quality, supported by actual data and results from some game simulations. Furthermore, the report on our "learning journey" highlights the strengths and weaknesses of our approach. It provides an analysis of what aspects were handled effectively and what could be improved if we were to revisit this project. This introspective examination aims to shed light on our experiential learning, mistakes made, and insights gained, offering a comprehensive view of our project's life cycle.

2 Heuristics

In this crucial section of the project, we delve into the core of our strategic decision-making process: the heuristics e1 and e2, which we meticulously developed for the AI Wargame. The essence of our project's "success" depended on these heuristics, as they formed the mainstay of our AI's decision-making algorithm.

This section will provide a detailed overview of both **e1** and **e2**, elucidating their functionalities, strategic implications, and the underlying logic that drives their decision-making processes. Furthermore, an essential part of this section is the comparative analysis of these heuristics. We will examine their performance in the tournament, assessing their strengths, weaknesses, and effectiveness, providing a comprehensive understanding of how each heuristic behaves under different game conditions.

Note that each aspect of the heuristics is given a specific weight, indicating its importance

in the AI's strategy. The use of these weighted factors allows the AI to dynamically adjust its strategy based on an assessment of the current game situation.

2.1 E1

Heuristic e1 operates as a nuanced summation of the disparities in various game parameters between the two players, each parameter weighted to signify its strategic importance. The heuristic encapsulates critical aspects, namely health difference, total health, total units, damage potential, and repair potential. This section expands on the explanation provided, emphasizing the underlying logic behind the heuristic's construction and its integration into the decision-making fabric of our game.

The heuristic accounts for five variables, each offering a unique perspective on the game state:

- AI Health: Represents the disparity in health between the two players.
- Total Health: Reflects the collective health sum of all units on the game board.
- Total Units: Indicates the total count of units present on the board.
- Damage Potential: Quantifies the potential offensive capabilities of both players. Evaluates the potential damage output from both players by considering the average damage inflicted by each unit type.
- Repair Potential: Estimates the restorative capacities possessed by each player by incorporating the average repair values associated with each unit type.

The process commences with the initialization of adjustable weights (w1, w2, w3, w4, w5) and the creation of a matrix, referred to as "*count*". This matrix serves as the numerical canvas upon which unit counts and health values are recorded. Iterating through the dimensions of the game board, the heuristic diligently counts units and accumulates health values. Each non-empty cell contributes to the count matrix, with health values recorded according to the unit's player and type.

As a summary, these are the main characteristics of our algorithm:

- Heuristic refinement: The e1 heuristic embodies a weighted sum of disparities across pivotal game parameters between players. Each parameter undergoes a multiplier effect, introducing a dynamic aspect to its significance. For instance, while every health point is a valuable metric, the heuristic acknowledges that the impact of a unit kill far surpasses an individual unit's health. This distinction is achieved by applying weight multipliers

that proportionally enhance or diminish the contribution of each parameter to the final heuristic value.

- Unit Type Differentiation: A distinguishing feature of e1 lies in its treatment of unit types. Recognizing that not all units are created equal, the heuristic incorporates separate values for damage potential and repair potential based on unit types. This granular approach ensures that the death of a unit carries a distinct weight, depending on its type. Consequently, the heuristic is finely attuned to the strategic implications of losing or damaging units with varying capabilities, offering a nuanced evaluation of the game state.
- AI priority: Acknowledging the pivotal role of the AI in the game dynamics, e1 designates a specific weight, denoted as "w1", exclusively for AI-related considerations. This deliberate allocation underscores the strategic importance of AI decisions within the heuristic framework, ensuring that every single health point on the AI influences appropriately the decision-making process.

Set in a formula, the heuristic looks like this:

$$e_1 = (10000 * AIH) + (1 * TH) + (10 * TU) + (2 * DP) + (2 * RP) \quad (1)$$

where
AI = AI health;
TH = Total Health;
TU = Total Units;
DP = Damage Potential;
RP = Repair Potential

2.2 E2

In contrast, **heuristic e2** took a different approach, emphasizing on variables along the lines of:

- Damage Potential: This calculates the potential damage output for each player based on the number and average damage of each unit type.
- Repair Potential: Similar to "Damage Potential", but for repair capabilities. It calculates the repair power for each player based on the number and average repair of each unit type.
- Number of turns: Constitutes the number of turns played. Its importance resides on the fact that as the game progresses, the importance of certain actions can change.

- Weighted Branching Factor: Calculated as the average of evaluation per depth in the game tree, it was included because a higher branching factor means more possible moves, indicating a more complex game state. The AI might use this information to adjust its search depth or strategy.
- AI's Health: It represents the AI's health difference. The main reason why we considered it is because it is a direct measure of the AI's survivability in the game.

By implementing **dynamic scaling with turn count**, we introduce adaptability into the heuristic's sensitivity to specific parameters over time, e2 heuristic incorporates a mechanism linked to the turn count. By multiplying the designated weight ("*w3*") with *self.turns_played*, the heuristic accommodates an evolving strategic landscape. This dynamic adjustment ensures that certain parameters can gain or lose importance as the game progresses, aligning the heuristic's sensitivity with the evolving dynamics of the game scenario. Moreover, we allow the evaluation function to give more importance to simpler states (with lower branching factors).

In games where strategies might change over time, the **number of turns** is crucial. By considering them, the heuristic can be adjusted to favor positions that lead to faster victories or positions that ensure long-term sustainability. For example, we may want to prioritize aggressive strategies in the early game and defensive strategies in the late states. This aims to emulate the type of "meta-game" decisions a human player might make.

All set in a formula where the decision-making process is composed by immediate game state factors and strategic implications, the heuristic would look like this:

$$e_2 = (1.2 * DP) + (1.2 * RP) + (1.1 * T) + (1.05 * WBF) + (10000 * AIH) \tag{2}$$

where
DP = Damage Potential;
RP = Repair Potential;
T = Number of Turns;
WBF = Weighted Branching Factor;
AIH = AI Health

2.3 Comparison of heuristics

Both e1 and e2 heuristics share a common goal of evaluating game parameters, considering damage potential, repair potential and AI health in both cases.

However, e1 extends its evaluation to encompass total health and total units, offering a more comprehensive perspective on the game state. In contrast, e2 introduces adaptive weighting

and dynamic scaling with the number of turns and the weighted branching factor, enhancing its adaptability to evolving game scenarios. The choice between e1 and e2 depends on project requirements and the desired balance between granularity and adaptability. Thorough testing, tuning, and documentation are essential for optimizing the performance of either heuristic in diverse gameplay situations.

This was glaringly obvious during the tournament where the defender chose to suicide one of its units, inexplicably, before proceeding to play normally. As an Attacker, on the other hand, every move was clearly motivated towards the victory condition. Because the board is asymmetric it would have been a logical strategy to make an e1 dedicated to the attacker and an e2 dedicated to the defender instead of making 2 symmetric evaluators of different styles. But our solution, within that limited sample, performed well.

3 Alpha-beta pruning efficiency

Minimax is a recursive algorithm that tests successive moves to evaluate how good the outcome is. It successively maximizes the gains of the target and minimizes those of the opponent. We model this as a depth-first tree traversal. Adding alpha-beta pruning is an optimization where parts of the traversal are stopped when a more favorable score has already been found at a particular node. In practice we found that this cut the evaluation time at equal depth to about 1/10 the evaluation without pruning.

The actual theoretical difference between the two depends on the order the moves are tested in. With 6 units each being able to act in 2 or 4 directions, the branching for the average node is 18 children, which means minimax alone would take $O(18^{depth})$, while the pruning divides the depth exponent by a factor that depends on the order mentioned above $O(18^{depth/n})$. This n will be a value between 1 and 2 (1 is worst case scenario with no improvements over minimax and 2 is best case scenario with perfect ordering). It is in direct relation to the amount of branches pruned.

We aren't making any attempts at ordering the moves so it's safe to assume our gains are actually quite small. We can even evaluate that the pruning is only a factor of around 1.25 in our execution for the 10x speed improvement we see (see figure 1 and 2). And yet that reduces the number of evaluations made to less than 1% in comparison with the unpruned traversal. *It is likely this evaluation, even thought mathematically sound, is very wrong in practice because we don't know how much of the execution time is attributed to different aspects of the computation.*

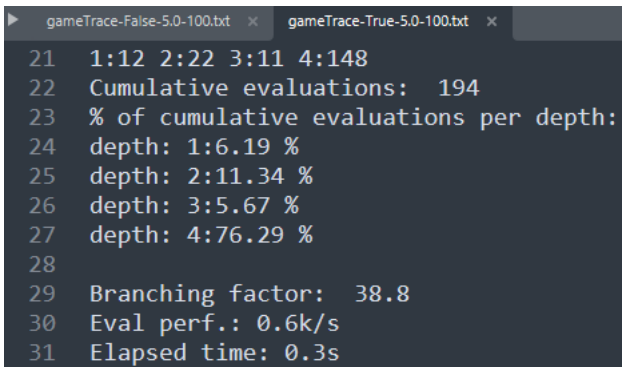

Figure 1: Pruned

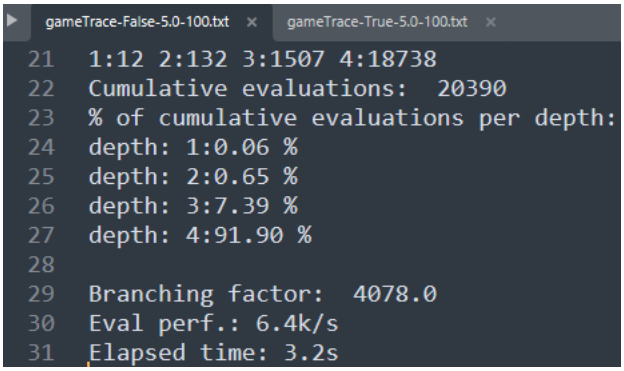

Figure 2: Unpruned

In fact our implementation could be improved further by coming up with an ordering schema that produces more pruning. We could also stop resorting to deep copy of the entire game object for the algorithm to evaluate on. If the recursion only handled moves by adding an "undo_move" method to call when recursing back instead of operating every evaluation on a complete deep copy of the game state, the execution would be orders of magnitude more efficient. However, the code complexity required and quality assurance is much higher so we made the trade-off given our limited time to use this less efficient less error prone approach.

4 Lessons learnt and Project Review

In retrospect, the project presented a solid foundation with successful elements such as the implementation of heuristics (e1 and e2) that effectively considered key game parameters for decision-making. The inclusion of variables like damage potential, repair potential, and AI health contributed to a robust strategic framework.

However, there were opportunities for improvement, particularly in the complexity of the algorithms and the necessity for meticulous tuning. In future improvements, streamlining the heuristics and optimizing their performance across diverse scenarios would be a priority. Additionally, a more detailed documentation process and thorough testing would enhance the clarity and effectiveness of the implemented strategies.

Other paths of improvement would be to test strategies such as running an AI model to determine the best weight values for each heuristic or, as the first moves are usually similar from one game to the next one, we may explore which ones are better and set them as predefined in the first turns to save computation time.

Concerning the team's dynamics, effective communication and collaboration were fundamental in navigating challenges, especially concerning D2 (which was the most challenging assignment for us). In future endeavors, focusing on regular progress updates would enhance

the overall group productivity.

5 Conclusions

In conclusion, our project successfully implemented an adversarial search algorithm for the AI Wargame, featuring heuristics that evaluated key game parameters. The tournament performance highlighted the adaptability and effectiveness of the solution, and the report delves into the efficiency of alpha-beta pruning, showcasing a notable reduction in evaluation time.

While acknowledging the project's strengths, we recognize opportunities for improvement, particularly in algorithm complexity and heuristic tuning. Overall, the project provided valuable experiential learning, shedding light on both successes and areas for enhancement in our strategic decision-making process.