COMP424 Final Project Report

Hanwen Wang (#260778557)

Hao Shu (#260776361)

1. Introduction

The aim of this project is to design an AI player for the game called Saboteur, where each player aims to get the nugget in the hidden tile to win the game. The AI agent will receive the information on the current board state, and choose his move based on this information.

This report will be briefly talking about the strategies of winning the game, analyzing the pros and cons, the reason why we implement the algorithm we've chosen, and possible improvements in the future.

2. Current Approach and Motivation

2.1. Overview

In this project, we use the knowledge related to Constraint Satisfaction Problems and Planning. Since we will have many different board states in different games, with different cards in hand and in the deck, we have to get the current status and then consider our moves depending on the cards we have.

2.2. Constructive Approach

The constructive approach is one of the two approaches for solving constraint satisfaction problems, and we use this method to solve various situations we might face during the game. This method contains 4 aspects. The first aspect is the states, it is defined by a set of values assigned so far, which in this case is the current state of board and card in hand. The second aspect is the initial state, which in this case is the original board state and card in hand at first.

The third aspect is operators, it assigns a value to an unsigned variable, which in this case is processing a move. The last aspect is the goal test, all variables assigned and no constraint violated, which in this case is the game is over and no illegal moves exist. In the following paragraph, we discuss our main logic of facing different constraints. First, we need to check whether we can use a tile card or not. If not, we need to check if we have a bonus card in our hand to cancel this malus effect. If we still do not have it, then we have only two options for our next move. The first option is to drop a card while the second one is to use a function card. However, if we can use a tile card at first, we need to check if we know the nugget position. If we know it, we then use the method of state-space planning, which we will discuss in the following paragraph. If we do not know the nugget position, we check our hand for the map card first. If we do not have it, we have three options this time. The first and the second options are the same as the ones in the situation discussed above, while the third option is to process the move assuming the nugget position is in the middle of three hidden tiles. These are the general ideas of choosing the next move depending on the current state.

2.3. State-space Planning

State-space planning is one of the two approaches to planning, it works at the level of states and operators. It is the process of deciding which parts of the state space the program will search assuming the universe of possible solutions to be searched is called the state space. In this case, all legal moves are the state space and our search or decision functions are the process. In our project and for every move, we need to decide which card to play and which position to play. Our general idea of dropping a card is that drop the blind alley card in order to save more useful cards in our hands. And if we know the nugget position, we can also drop the map card since it is no longer needed. To process the next move, our idea is to choose the tile, which can form a path, that is closest to the nugget position among all legal moves. This approach can help us move towards the destination, instead of moving randomly. These are the two ideas of processing a move, we aim to keep useful cards in hand while moving towards the nugget position.

2.4 Motivation

The reason why we choose these two methods is that there are so many possibilities of the next legal move and uncontrollable factors like the cards in hand, current board status and opponent's move. As a result, we have to take into consideration as many situations as we can think of and find a solution to the next move to the corresponding situation, then we need to move to the target tile in a greedy way.

3. Analysis

3.1. Advantages

- Our agent can make sure it moves towards the right target tile.
- Our agent always chooses the move to shorten the distance between the starting point and the target position if it chooses to play a tile card on the board.
- Our agent can make sure it is able to correctly use the function card when wanted.

3.2. Disadvantages

- If the card path is connected but the int path is not connected, our agent can detect the wrong tile and destroy it, but it does not have the ability to correct it.
- Our agent can only choose the move that is moving towards the target. It does not have the ability to fix the path if the path is destroyed by others in the middle.
- Our agent does not have the function of calculating the number of cards of a specific type played or remained.
- Our agent cannot make sure it has enough correct cards to play consecutively to win, e.g. it takes two steps to win, but our agent might only have one correct card to play, it will still play that card.
- Our agent does not have the ability to predict other player's moves and respond to those moves.

4. Other Approach Tried

Minimax algorithm:

According to the naive approach of the minimax algorithm, the main idea of the alpha-beta pruning is that we prune some of the game options from the search tree by following the logic that both players in the game aim to win in a greedy way. However, the algorithm only produces a similar result with greater search time due to the searching with large depth.

Monte-Carlo tree search algorithm:

Considering the large branch factor that Saboteur has, we also tried the Monte Carlo Search approach. It first selects the best node based on the UCT value. Then it expands the node and performs simulation all the way to the end of the game [1]. It backtracks and updates the win rate of all the parent nodes in the end. The whole procedure is repeated as many times as possible to get a better result. The final move is selected based on the win rate. During our implementation, the search is able to visit a lot of nodes at the first depth, but as the depth increases it is less likely to simulate many times on children nodes. As a result, the algorithm is not optimal to select the best move, but reach the sub-optimal move for the most of the time.

5. Future Improvement

- **Implement Monte-Carlo tree search with the Minimax algorithm:**
  Although the Monte-Carlo tree search does not give us a very optimal solution, it can help us reduce the number of moves that we need to consider. Since there are random moves involved in the Monte-Carlo algorithm, it helps with not losing the possibilities to win in case the opponent of the agent does not move very greedy [1]. Also, the feature of the Monte-Carlo algorithm makes it an ideal complement for the Minimax algorithm, because it focuses on one path and makes the AI agent less

time-consuming. In addition, MCTS with Minimax can easily be extended and modified to work in more complex environments [1].

- **Means-Ends Analysis**:

  As Krisnadi Dion et al. stated in the paper “Decision System Modelling for "Saboteur" using Heuristics and Means-Ends Analysis” [2], Means-Ends Analysis could give a more optimal solution for the Saboteur AI agent: Means-end analysis incorporates a heuristic value in each card. The value will change according to the role of each player and where this card is played. This value will then be used in determining the role of a player who made a certain move, and the best move for a certain player [3]. The agent using the Means-Ends approach could make predictions on other player’s move and achieve a maximum winning rate of 98.2% [2]. And we could improve our agent by implementing the Means-Ends Analysis for its outstanding performance in Saboteur.

Reference:

[1] Muens, P. (2020). Game AIs with Minimax and Monte Carlo Tree Search [Online]. Available: https://towardsdatascience.com/game-ais-with-minimax-and-monte-carlo-tree-search-af2a177361b0

[2] Krisnadi, Dion & Lukas, Samuel & Logan, Laurentius & Bachtiar, Nadya & Lohanda, Livia. (2019). Decision System Modelling for "Saboteur" using Heuristics and Means-Ends Analysis. 178-182. 10.1145/3369114.3369162.

[3] D. Krisnadi and S. Lukas, "Saboteur game modelling using means-ends analysis," 2016 International Conference on Informatics and Computing (ICIC), Mataram, 2016, pp. 319-324.