COMP 30024 - Project B Report

Tianyang Chen(1089194), Dan Wu(926444)

ALGORITHMIC APPROACH

Ø Actions

Generally, we use minimax algorithm and alpha-beta pruning for our important techniques for deciding on which actions to take throughout the game.

Minimax algorithm and alpha-beta pruning are normal ways to implement board games and easy to use for us. The Minimax algorithm is often used in games and programs, such as chess, in which two players take turns, one step at a time. It's a zero-sum Algorithm, in which one side has to choose the option that maximizes its advantage, and the other side has to choose the option that minimizes its opponent's advantage.

Minimax is a pessimistic algorithm, which assumes that every move the opponent makes will lead us in the direction of the pattern, which is theoretically the least valuable at the moment, that the opponent has the perfect decision-making ability. So our strategy should be to choose what the other side can achieve to make our worst-case scenario the best, which is to make the other side in a perfect decision to cause me the least loss.

Our program first generates all legal actions including Move actions and Boom actions, then the minimax algorithm randomly selects one of them. After that, the algorithm checks if the Boom action is meaningful. A meaningful Boom action means if the token is exploded, the opponent tokens will be exploded. If this action is meaningless then it will be skipped. Finally, the approved action is applied to the iteration. This step reduces alpha-beta pruning's work by removing some useless branches in advance.

The biggest disadvantage of the minimax algorithm is data redundancy, which has two situations: ① maximum redundancy; ② minimum redundancy. Correspondingly, alpha pruning is used to solve the problem of maximum redundancy, while beta pruning is used to solve the problem of minimum redundancy, which constitutes a complete alpha beta pruning algorithm.

For search depth, we have planned to implement a dynamic depth, but it is hard for us to complete. During the development, we just adjusted it manually to control the speed of generating new actions.

Ø Evaluation

Evaluation function is an important part of this algorithm. A good evaluation function can not only help win the game, but also help prune and improve the search efficiency.

We have two plans for the evaluation part.

The first plan

The heuristic value of the state consists of three parts, N, G and D.Where N is the current token number, g is the Group number, and D is the average distance between each token.

Heuristic value is a weighted sum of the three, with weights of 100, 10, and 1, respectively. These coefficients are decided based on experiments during the development of the program. N has the highest weight because, according to the winning logic, only the remaining token wins, so the part with more tokens has a better chance of winning. On this basis, G takes second place, because with the same number of Tokens, the more group number, the less likely it is to be hit by boom. Finally, we added the most heuristic part of the average distance between each token, which is that we want to have as much distance as possible between tokens in each group in order to form a new group.

The second plan

Generally, during the evaluation, we calculate the overall score by merging the score of the initial colour and the score of the opponent. The score will be increased if the state is favorable to the initial colour, or decreased if the state is unfavorable to it. On the contrary, the score will be decreased if it is favorable to the opponent, or increased if the state is unfavorable to it.

All tokens are divided into three kinds - common tokens, harmful tokens and threatening tokens. Common token means if it conducts an active explosion, no other tokens will be exploded. Harmful token means if it conducts an active explosion, no opposite tokens will be exploded, while other tokens of this side will be exploded. Threatening tokens means if it conducts an active explosion, it will explode the opposite tokens. From the

explanation, it should be concluded that a token can be both a harmful token and a threatening token because its explosion can lead to two sides' tokens being exploded.

As for a common token, its basic score is +10. Normally, N common tokens will be given N * 10 scores. However, a common stack with N tokens will be given k * N * 10 scores (k>1) because tokens in a stack have more options.

As for a harmful token, its basic score is -20. Normally, N harmful tokens will be given N * -20 scores. However, a harmful stack with N (N>1) tokens will be given k * N * -20 scores (k>1) because more damage is undertaken by this side (lost more execution opportunities at once). And the more tokens of this side are exploded, the more score is reduced for this token.

As for a threatening token, its basic score is +20. Normally, N threatening tokens will be given N * +20 scores. However, a threatening stack with N (N>1) tokens will be given k * N * 20 scores (k>1) because more damage is undertaken by this side (Lost more execution opportunities at once). And the more tokens of the opposite side are exploded, the more score is increased for this token.

For each side, sum up the results of their own tokens.

Apply the method above to measure tokens of the initial colour and the opponent colour and calculate the result respectively.

When calculating the total result, the result of the initial colour * (+ 1) will be added to the total result, and the result of the opponent colour * (- 1) will be added to the total result.

However, due to the limitation of time and our personal python skills, we only realized the first plan successfully.

OVERALL EFFECTIVENESS

Basically, this program can be successfully used in Expendibots games, although there will be some behaviors that seem to be not intelligent enough from a human's point of view. It is expected to defeat opponents of low-level difficulty and have a chance of defeating opponents of middle-level difficulty.