ハノイ国家大学
Nguyen Phuong Linh
Hoang Van Phu
Nguyen Hoai Son（教員）

I. Introduction

Our program is a Web application, which is divided into two parts: Client-side program and Server-side program. Client-side program includes a user interface which allows the Tower to update the game's state easily, while Server-side program provides the Tower suggestions of next moving steps.

II. Client-side program

We used HTML, CSS and JavaScript to write the Client-side program. It provides the Tower with several tasks such as import of competition field data, updating two teams' move, calculating 2 teams' score and competition field data backup for recovering from program failure. Our Client-side program can help the Tower to grasp 2 teams' situation in order to decide what strategy should be used.

III. Server-side program

The Server Application was written in C++ and is run as a service on Web server. Whenever the status of two teams are updated by Client-side program, the Server-side program will calculate the best solution for the next moving steps. In order to do that, we use Minimax algorithm and Alpha - Beta pruning algorithm. The descriptions of the algorithms are as follows.

3.1 Minimax algorithm

Minimax is a kind of backtracking algorithms that is used in decision making and game theory to find the optimal move for a player, assuming that your opponent also plays optimally. Thus, we use this algorithm to build a game tree which leads to all states available after n steps from current state. Our program then searches through the tree to find out the best solution which can give the highest profit.

However, there are too many states that need to be searched in the tree. Hence, it takes our program a lot of time to process and exhausts computer memory. To reduce the calculation cost and memory, we use Alpha – Beta pruning algorithm.

3.2 Alpha – Beta pruning algorithm

Alpha – Beta pruning is a searching algorithm that try to decrease the number of nodes that need to be evaluated by the minimax algorithm in its game tree. It cuts off branches in the game tree which need not searching because there already exists a better node available. By using this algorithm, our program does not only search faster but also goes into deeper levels of the game tree. Therefore, our strategy is able to find the best solution for next moving steps within given time.