Design and Implementation of Common Platform Based on UCT Algorithm

Shu-qin Li¹,²*, Yao Fu¹ and Ya-nan Qi¹

¹School of Computer, Beijing Information & Science and Technology University, China
²Sensing & Computational Intelligence Joint Lab, Beijing Information Science & Technology University, Beijing 100101, China

*Corresponding author

Keywords: Computer chess game, UCT algorithm, Common platform, Java.

Abstract. With the development of computer chess games, now most of computer chesses have implemented the UCT search algorithm accordingly. And we have verified the superiority of the algorithm in experiments of some chesses. However, UCT algorithm is complex in programming and takes a lot of effort in development and tuning. This paper presents a common platform based on UCT algorithm suitable for computer chess game, which can help save time of development and achieve automated tuning. It achieves good result in both experiment and practice.

Introduction

Computer chess game is considered as one of the most challenging research directions in the field of artificial intelligence [1]. It is also an important area for the development of artificial intelligence and has shown great potential in recent years. Developer programs to achieve function of playing chess automatically, so programming is an important process in computer chess game. A computer chess game program consists of several major functional modules: a board representation and reading module, a legal expression module, a search module and an evaluation module. The search module and evaluation module are the core parts of the whole program. The quality of code in these parts directly determines the chess power of the program.

UCT algorithm [2] is upper bound confidence interval algorithm applied to game search tree. This emerging search algorithm uses combination of depth and breadth search based on upper bound confidence value of UCT formula, which has the advantages of time and space compared with the traditional search algorithm in the search process of large-scale game tree. It is possible to get a global best solution to current situation within controllable time based on combination with a risk evaluation function.

Many chesses such as Go, Amazon chess, flag and so on, they’ll make a huge game tree when the game is in progress, which brings a great challenge to search module of computer chess game program. Traditional search algorithms such as maximum-minimum search algorithm, Alpha-Beta pruning algorithm, etc. are not suitable for huge branch tree search. So, we see programs with traditional search algorithms always have weak chess power. However, because of the complexity in development and tuning of UCT search algorithm, most time will be wasted on programming. Therefore, we design and implement a common platform based on UCT algorithm, which can help speed up development of search module and make simple tuning, which greatly reduces the difficulty of programming.

Design of Common Platform

There are four important parameters in UCT algorithm, namely, depth (layer), breadth (number of nodes), adjustment parameters α, condition of end. The condition of end includes limited time and limited depth. We develop the common platform by using object-oriented programming language Java, which shield specific platform environment requirements [3]. You can run Java programs on
multiple platforms so long as they support Java virtual machine. This is in line with common requirements of our platform.

In actual development, we hide unrelated functions using private encapsulation, and expose tunable parameters and related methods by API, so that user can quickly build UCT algorithm without unimportant technical details.

If user doesn’t have a good solution for UCT algorithm parameters at the beginning, our common platform can be used to automatically tune parameters by giving a parameter range, and finally select a best parameter in the range.

Through above analysis, the structure of the common platform is shown in Figure 1.

![Figure 1. Structure of common platform.](image)

The platform can be divided into the following sections according to the module:

Resource Management: This module mainly includes allocation and management of allocation and management of CPU and memory. This module will allocate resources reasonably based on hardware resources of user when execution of UCT algorithm and automatic tuning of parameters.

Execution Library of UCT Algorithm: When a user makes UCT search with common platform, algorithm execution library executes the algorithm, including search algorithm and proceed calculation, etc. Finally, a best move in the current situation will be returned by the executive library.

Tunable Parameters: All parameters which users can change are tunable parameters. For example, the depth, breadth and adjustment parameter of UCT algorithm. These parameters are got and set by function get and set.

Automatic Tuning: Users can specify a range of tunable parameters for simple automatic tuning. The platform uses Cartesian product of different elements in the designated range by users to make different-parameter computer chess game programs. The platform count rate of wining for every program by making them fight with each other, and the best parameters come from one with best rate.

Interface: Interface is actually a collection of common functions that expose several available functions of common platform to users. It shows major functions of the platform, such as the execution function of UCT algorithm, setting function of parameters and automatic tuning of executive function.

**Implementation of Common Platform**

In design of UCT algorithm, we consider generality of platform on different characteristics of chesses, different expressions of board and move by users. And the key to achieving generality lies in the choice of attribute types.

**Design of Node in UCT Search**

The node maintains storage of current board, move and \( a, M, \bar{m}, \bar{v} \) in formula of UCB. Figure 2 shows UML class of uctNode. We focus on function randomPlayof class uctNode, which is a recursive function. Each time the method is executed, it determines whether reach the end layer of depth, if not, if continue. The method generates all feasible moves in the current situation, if there isn’t move, exits; If there are feasible moves, it randomly selects a move to generate a new node under the game tree. Then it calls itself recursively until reaches limited depth.
Design of SearchClass of UCT Algorithm

Class uctSearch maintains a game tree to express all moves and branches that simulation generates. Figure3 shows UML class of uctSearch.

Function createLeafNode is designed to expand node horizontally in UCT algorithm, that is, expansion of the breadth of game tree. The method receives an integer array expan as number of breadth nodes whose subscripts represent number of layer (depth), elements represent number that every layer need extend to. Users can specify the number of nodes that UCT algorithm simulates on each layer of expansion by array expan.

Design and Implementation of Evaluation Function

There is no such concept of evaluation function in UCT algorithm [4]. However, because of the complex situation and the huge branches, problem of lacking breadth could be caused by forcing simulation to the end of a chess game. Therefore, with the limit of hardware resources, we need to give a relatively reasonable depth and use evaluation function to score nodes before the simulation can reach the end of a chess game. The platform provides two schemes for users: (1) Assign a search depth to UCT algorithm, use evaluation function to score nodes before the algorithm simulation process can reach the final situation, and use win (score 1) or fail (score 0) to score nodes when simulation can reach the end. Moreover, users need to specify depth, evaluation function explicitly. (2) Use default mode, the algorithm can only score nodes when simulation reaches the end. These two schemes will be introduced in section 4.

Design and Implement of Automatic Tuning

The platform enables the selection of the best for a given range of parameters. For example, if a user wants to find the optimal search depth within the range of [5, 10], the platform will create six game programs with search depth layers of 5, 6, 7, 8, 9 and 10 (for convenience, here is only one parameter as sample, the other parameters are in default value). The six game programs fight with each other by a designated number of games. The platform counts rate of winning for every program, and the optimal depth comes from one with best rate.

The method uses an input of map and stores all parameters in array param by iteration. Then it uses function of array to generate Cartesian product of all different parameter (like depth, breadth). And it creates several game programs by anonymous function, makes them fight with each other for designated times, finally it returns the best parameter(s).
Case Explanation

In this section, we'll explain use of our common platform and parameters tuning. In this case, we declare depth and evaluation function.

Precondition: We have finished expression of board and generation of moves, the board attribute is required named board and must be a two-dimensional Byte array; Moves must be expressed as two coordinate arrays with length 2, named preLoc and nowLoc, respectively. That is to say, a move consists of previous location and current location of a moved chess. Moreover, the classes in the program project need to be named Board.java and Move.java.

**Step 1:** Unzip the common platform to srcdictionary of the whole project. Then create a uctSearchinstance where the project needs to call search function like:

```java
uctSearch search = new uctSearch(evalFunc:evaluation(), depth:a, wide:b, alpha:c);
```

Before declaring a uct Search instance, the user must specify depth, breadth and adjustment parameter of search. Because of the provided depth, an evaluation function must be given. The evaluation function is transferred to constructor of the instance, and the function is required to return an evaluation which is a double attribute (if you use default mode, just transfer null to evalFunc, then any parameter transferred to depth will be ignored). In addition, all parameters transferred to constructor could be changed, by:

```java
search.forceSetEvalFunc(evalFunc:evaluation());
search.forceSetDepth(depth:a);
search.forceSetWide(depth:b);
search.forceSetAlpha(alpha:c);
search.forceSetTime(limitTime:d);
```

Focus the limited time set by forceSetTime, the time is in second.

**Step 2:** Then add code as at the start of search of the program.

```java
try{
    Move bestMove = search.run();
} catch(timeLimitedException ex){
    Move bestMove = ex.getLocalMove();
    ex.getMessage();
}
```

There the user needs to declare a Moveinstance named bestMoveto receive the current best move by search function. If the user has set limited time, search function will throw an exception which reminds the limit reached, and return local best move by function getLocalMove, then print a related message; If not, search function will reach the declared depth and breadth, then return a best move.

**Step 3:** If the user need use the platform to make a simple tuning for parameters, he should create an executable class additionally. Then declare a HandleParaminstance in the mainfunction, and call method getBestParamof this instance to get the best value over a range of parameters.

```java
public static void main(String[] args){
    HandleParam getDepth = new HandleParam(map[1,10], param:depth,100);
    HandleParam getWide = new HandleParam([6,15], param:wide,100);
    HandleParam getAlpha = new HandleParam([0,1], param:alpha,100);
    getDepth.getBestParam();
    getWide.getBestParam();
    getAlpha.getBestParam();
}
```
Note that the range of depth and breadth needs to be positive integers, while adjustment parameter between [0, 1]. It will consume large amount of computer resources because of the theory of tuning, so it’s not recommended for less-performing machines.

**Summary**

This paper describes the design and implementation of common platform based on UCT algorithm. UCT algorithm not only overcomes the shortages of traditional search algorithm but improves Monte Carlo simulation and reduces the calculation scale. The common platform modulates UCT algorithm so that users can save lots of time on programming by calling a few simple functions of execution library. And the platform can do simple tuning according to designated range of parameter given by the user. This make users focus on other more important aspects. We have proved the chess power by experiment and practice and has achieved good results. In the future, we’ll improve the tuning module and achieve fully automatic tuning based on resources of the user to make the platform more automatic.

**Acknowledgements**

This work is supported by 2017 Talent-Development Quality Enhancement Project of BISTU (5111723400), by Standard Project of Science Plan of Beijing Municipal Education Committee (KM201611232014), and by the special bidding project of teaching & education reform(2017JGZB08)

**Reference**


