Good morning, even one. My name is Ivan.

This morning I’am going to talk about Artificial Intelligence for positional games.

First I’ll give you some basic information about subject.

Both “Go” and “Dots” are positional games where stones and dots respectively are defined by its positions on rectangle or square field. The main goal of game “Go” is capturing stones, whereas the main goal of game “Dots” is capturing the dots.

I’ve divided my presentation into several parts.

First, I’d like to describe theoretical comparative analysis of games “Chess”, “Go” and “Dots”.

Then I focus on current methods of AI for game “Go”.

And finally I’d like to outline my current researches and further plan of works.

Now I will talk about them in more detail.

It is can be concluded from this table that:

1. The method of deep search in tree can be applied to chess games but cannot be applied for positional games due to huge branch factor.
2. Due to additive nature of game Go and Dots, the number of possible game states increasing during the time.
3. State representation and estimation function of positional games are not trivial unike the Chess. The value of a Go position depends on a complex analysis to determine whether or not the group is alive.
4. Human is playing in games “Go” and “Dots” very well due to static nature of positional games unlike games, such as “Chess”, “Reversi” where figures change location frequently.
5. There are many scientific works related to research of games “Chess” and “Go”, unlike game “Dots”.

Next I describe current methods of AI for positional games.

# The traditional method of finding best moves of any logical game is search in tree.

The usual minimax search is simple algorithm that generates possible opponent moves on each turn and recursive play next iteration until depth value is not zero. On the last iteration it returns evaluation function value for current game state. Next this algorithm maximize this value on every odd depth of tree and minimize it on every even depth of tree.

There are several techniques, which can greatly improve the performance of search trees in terms of both speed and memory. Alpha-Beta, Negascout and MTD-f include some optimizations for pruning useless branches without influence on final result. Usually they efficiency depends on move generation order.

Finally the advantages of this tree search methods are:

* It’s simple to implement this methods.

The disadvantages are:

* It’s hard to create efficient evaluation function.
* It’s hard to create efficient moves generator.
* Huge branch factor in the tree.
* It’s hard to implement efficient parallel algorithm.

It’s well known that many collisions in game tree are occurred in such algorithms. Caching is used for this collisions minimization.

Zobrist hashing is very popular in Go and Chess programs because it has low collision rates, and can be iteratively updated at each move with just two XORs.

After Zobrist hash key has been calculated, it stored to common transportation table for further using in tree search algorithms.

**Advantages of caching:**

Game search tree collision minimization. Best for iterative algorithms.

Caching can be used not only for sequence of putted stones or dots but for group of stones and dots too.

Caching can be used in further calculations.

**Disadvantages of caching:**

Hard to determine errors are occurred.

Hard to realize tree parallel search algorithms with caching. Hence spinlock or atomic operations must be used.

# The second method of AI for positional games I would talk about is knowledge-based algorithms.

These algorithms based on idea of using expert knowledge’s, which contains in program. Common technics are pattern matching and pattern recognition.

Pattern matching algorithm is simple procedure of matching every pattern in knowledge base with all combinations of stones on game field. Some optimizations can be implemented.

Pattern recognition algorithm is procedure of recognition of pattern starting from position. Pattern recognition is based on idea of Define Finite Automate.

The advantages of these methods are:

* High performance.
* Global search.

Disadvantages:

* Requires many experts or professional players for building knowledge base.
* Hard to formalize expert knowledge.
* Requires much time to build knowledge base.
* Impossibility for deep search.

# And finale method of AI is Monte-Carlo Search.

This method is done by generating a list of potential moves, and for each move playing out thousands of games at random on the resulting board. The move which leads to the best set of random games for the current player is chosen as the best move.

In 2006, a new search technique, upper confidence bounds applied to trees (UCT), was developed and applied to many 9x9 Monte-Carlo Go programs with excellent results. UCT uses the results of the play outs collected so far to guide the search along the more successful lines of play, while still allowing alternative lines to be explored.

The advantages of these methods are:

* Easy to implement.
* Simple and efficient to realize parallel algorithm.
* Strategically search.
* Many high-level go engines are using UCT.

Disadvantages:

* Requires large computational resources for a good game.
* Best results for small games. But efficiency decreasing with board size increasing.

# Next I’ll talk about current researches

Simple move generator is used in experiments: included all empty positions with distance of <= 2 from all putted dots.

* I’ve been tested dependency of hash table size and number of calculating moves for alpha-beta algorithm with some heuristics.

From this dependency I made conclusion that the most efficient table size is two in power twenty elements and further size increase does not reduce the number of calculating moves.

* I’ve been compared UCT Method and Alpha-Beta search with some heuristics for game dots. UCT showed the best results than Alpha-beta. I conclude that the main problem of UCT is case of cascade attack-defense moves (also called «ladder»).

# Finally I want to describe further plan of works in next term

I would like to develop acceptable AI for positional game “Dots” in the next term. I would like to realize ideas displayed on this slide.

Tree search:

* The dots is statical game. Hence there is possibility to modeling sequence of moves with one color and finding preliminary alpha-beta bounds, which further are using in traditional tree search algorithms (Alpha-Beta or Negascout).
* Modeling of cascade attack-defense moves sequence.
  + It is a frequent case.
  + It can help to solve some problems with UCT method.
* Knowledge-based systems:
* Building pattern knowledge base for dots.
* Pattern recognition in dots with DFA.

Monte-Carlo methods:

* Calculating not all move sequence until the end, but calculating local sectors (such as group connections).
* «Smart» move generator (with patterns).
* Research UCT method with different parameters.

General:

* Trying to develop method of dividing set of dots on disjoint trajectories.
* Realize and explore all methods with combination of each other and different parameters.