Trabalho 1 - parte 1
INE5430 - Inteligência Artificial

Bruno Marques do Nascimento* Johann Westphall †

Florianópolis, 22 de Agosto de 2017

Introdução

Este relatório possui o objetivo de explicar o que foi desenvolvido para a primeira entrega do Trabalho 1 da disiciplina de Inteligência Artifical, ministrada pelo professor Elder Rizzon Santos na Universidade Federal de Santa Catarina(UFSC), campus Florianópolis. O propósito do trabalho é implementar o algoritmo de busca adversária MiniMax com podas α e β. A implementação será testada através do jogo 5 em linha (Gomoku), com tabuleiro tamanho 15x15.

Nesta primeira etapa, serão abordadas as definições matemáticas da função utilidade e heurística, e a implementação parcial da função heurística. Além disso, será explicado como são realizadas as detecções de fim de jogo e sequências de 4 peças.

A linguagem de programação *C++* foi a escolhida para a implementação do trabalho e para o desenvolvimento da interface gráfica a API gtkmm-3.0.

1 Definições matemáticas

1.1 Utilidade e Heurística

Os números abaixo foram gerados a partir de certos preenchimentos do tabuleiro.

Para o número máximo de unidades, foi suposto um tabuleiro com jogadas intercaladas. Para o número máximo de duplas, triplas e quádruplas foram supostos quadrados de tamanho respectivamente 2, 3 e 4, a Figura 1 mostra essa estimação para o número máximo de duplas. Foram contados o número de sequências formadas e os valores obtidos foram incrementados por uma certa margem de garantia com o intuito de evitar que uma disposição otimizada das peças no tabuleiro faça com que uma jogada de nível superior tenha menos peso que o máximo de jogadas de um nível inferior, por exemplo, uma dupla contará mais que o máximo de jogadas unitárias.

*brunomn95@gmail.com
†johannwestphall@gmail.com

Figura 1 – Exemplo estimação número máximo de jogadas.


```
n_max_unidades = 113
n_max_aprox_duplas = 100
n_max_aprox_triplas = 96
n_max_aprox_quadruplas = 70

nota =
   (n_unidades +
    n_max_unidades * n_duplas +
    n_max_unidades * n_max_aprox_duplas * n_triplas +
    n_max_unidades * n_max_aprox_duplas * n_max_aprox_triplas * n_quadruplas +
    n_max_unidades * n_max_aprox_duplas * n_max_aprox_triplas *
                                    n_max_aprox_quadruplas * n_quintuplas) -
   (n_unidades_adversario +
    n_max_unidades * n_duplas_adversario +
    n_max_unidades * n_max_aprox_duplas * n_triplas_adversario +
    n_max_unidades * n_max_aprox_duplas * n_max_aprox_triplas *
                                                    n_quadruplas_adversario +
    n_max_unidades * n_max_aprox_duplas * n_max_aprox_triplas *
                              n_max_aprox_quadruplas * n_quintuplas_adversario)

UTILIDADE e HEURÍSTICA = 1/n_jogadas * nota
```

2 Detecções

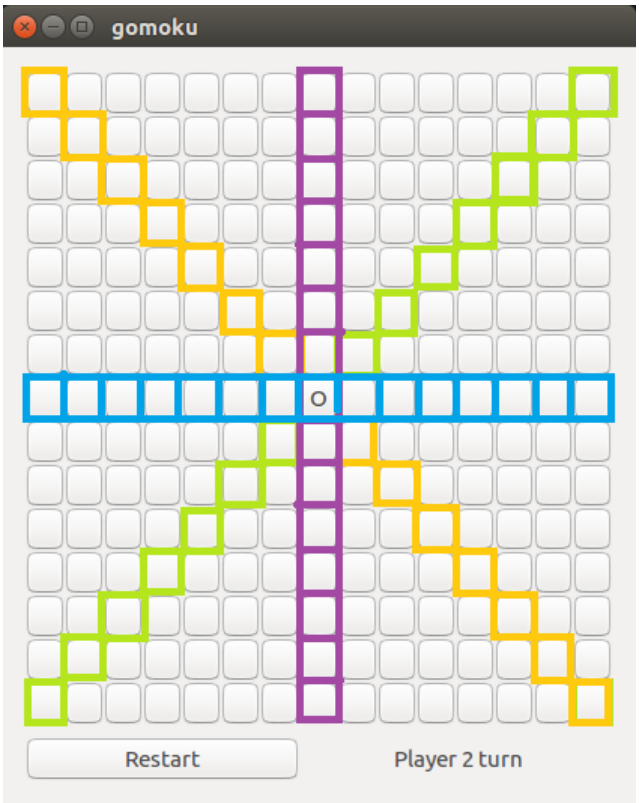
2.1 Fim de jogo

Após cada jogada feita por qualquer jogador é verificado se o jogo acabou a partir das coordenadas 'x' e 'y' da jogada. Considerando que uma sequência de vitória é formada por 5 símbolos consecutivos em qualquer direção, uma solução seria buscar no tabuleiro inteiro por essa sequência.

Todavia uma maneira mais eficiente de se fazer isso foi implementada. A partir do 'x' e 'y' da jogada, são extraídas as respectivas linhas, colunas e diagonais a qual esta jogada pertence, conforme Figura 2. Após, é realizada a busca em cada uma das estruturas extraídas com o intuíto de achar uma sequência que conceda a vitória para um dos jogadores, que seria uma sequência ininterrupta de 5 símbolos iguais, petencentes ao mesmo jogador.

A implementação deste algoritmo pode ser encontrada no arquivo *gomoku_core.cc*, nos métodos *have_winner()* e *find_sequence()*.

Figura 2 – Extração das estruturas pertencentes a jogada.



2.2 Sequência de 4 símbolos

A fim de detectar a sequência de 4 é aplicado o mesmo algoritmo para a detecção de fim de jogo, no qual a sequência buscada é de 5 símbolos, porém agora para um conjunto de 4 símbolos iguais consecutivos.

Foi verificado que esse método não detecta possibilidade com 'buracos' entre peças

como, por exemplo na Figura 4, e para a entrega final será implementado o algoritmo que verifica qualquer configuração de 4 símbolos que dê a possibilidade de vitória para um dos jogadores.

A implementação deste algoritmo pode ser encontrada no arquivo *gomoku_core.cc*, nos métodos *sequence_of_four()* e *find_sequence()*.

Figura 3 – Sequência de 4 símbolos: detecta.

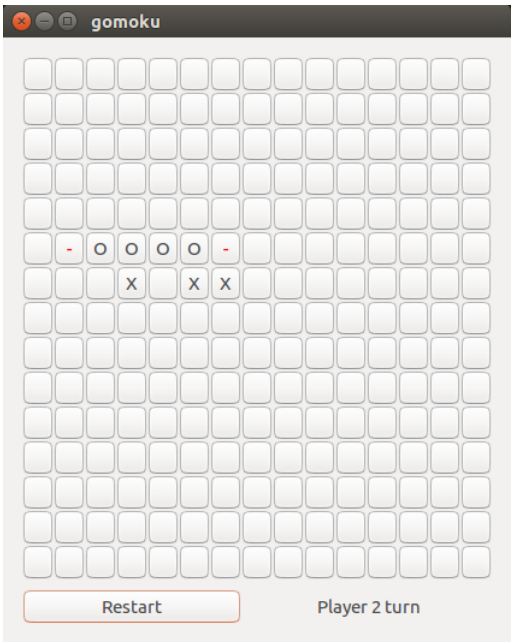


Figura 4 – Sequência de 4 símbolos: não detecta.