Universidade Federal de Minas Gerais
Inteligência Artificial para Jogos
Trabalho Prático 1 - Reversi
Documentação

Leandro Soriano Marcolino

11 de setembro de 2006

Sumário

1 Introdução

As árvores de pesquisa são muito utilizadas no desenvolvimento da inteligência artificial dos jogos clássicos, como dama ou xadrez. Um dos algoritmos mais importantes dessa área é o Minimax, onde o computador explora as possibilidades considerando sempre que o adversário irá jogar da melhor maneira possível.

Nesse trabalho iremos implementar o Reversi[1], um jogo clássico de tabuleiro, utilizando o Minimax para construir a inteligência artificial do jogo. Devido às limitações de processamento iremos utilizar uma técnica conhecida como poda Alfa-Beta (que evita a exploração de partes desnecessárias da árvore) e vamos limitar a profundidade da árvore que o computador irá pesquisar, utilizando uma função de avaliação nas posições limites.

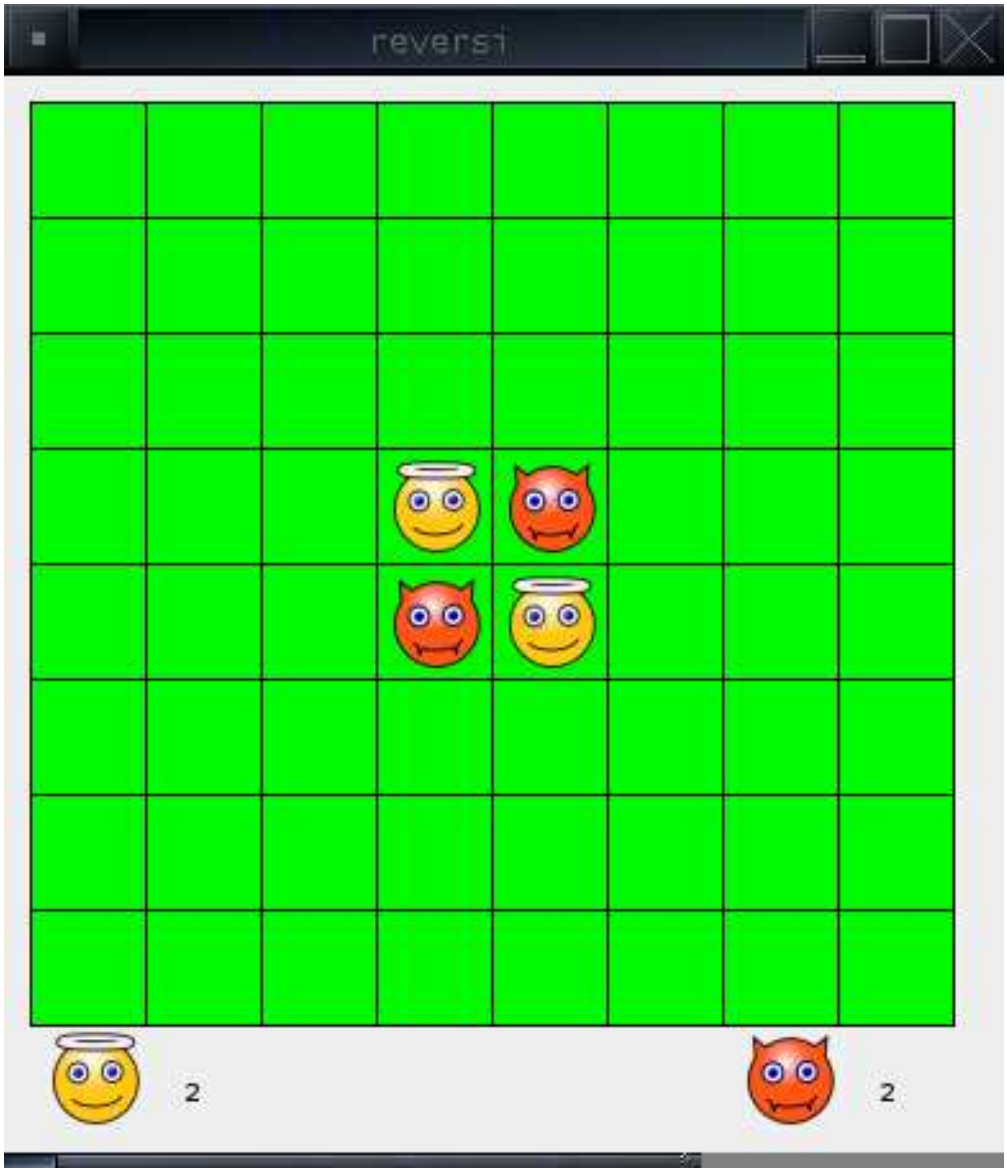
2 Como Usar o Programa

As instruções de compilação estão no arquivo Readme.txt. Após compilar o jogo, basta digitar para abrir o programa:

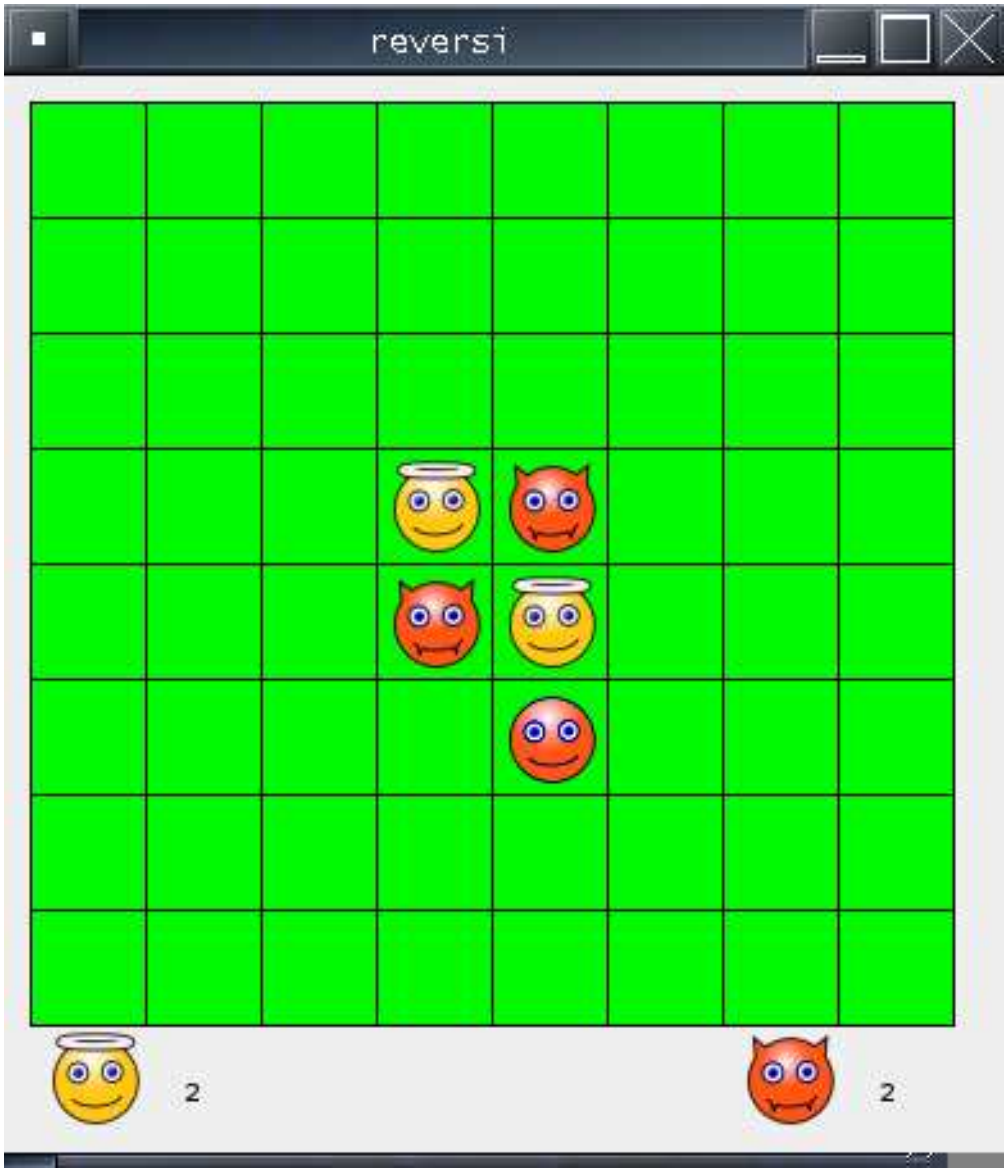
```
./reversi
```

Uma caixa de diálogo irá abrir, perguntando se você deseja jogar contra o computador ou contra uma pessoa. Caso deseje jogar contra o computador, irá aparecer outra caixa perguntando com qual peça você quer jogar e em seguida uma perguntando qual será o estilo de jogo do computador: agressivo ou estratégico. Esses estilos serão melhor explicados na seção Função de Avaliação.

Aparecerá então o tabuleiro, como pode ser visto na figura abaixo:



Para jogar basta clicar na casa correspondente. Aparecerá uma figura nas casas onde a jogada é válida, como pode ser visto abaixo:



3 Arquitetura do sistema

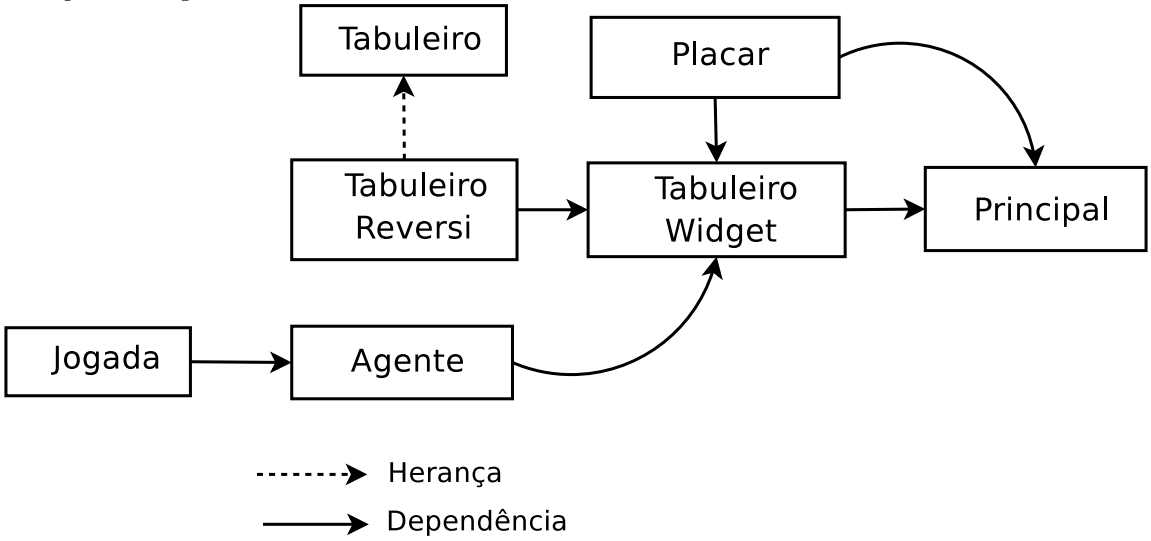
O sistema foi desenvolvido em C++ buscando seguir os princípios da Orientação à Objetos. Usamos a biblioteca QT para a interface gráfica. A avaliação do jogo feita pelo agente acontece em uma thread separada.

3.1 Módulos utilizados

- principal -> Abre o programa e inicia o QT.
- tabuleiro -> Cria um tabuleiro padrão.

- tabuleiroReversi -> Especializa o tabuleiro para um jogo de Reversi.
- tabuleiroWidget -> Cria o Widget (objeto gráfico) onde será desenvolvido o jogo e faz o controle do andamento do jogo
- placar -> Cria o placar do tabuleiro
- agente -> Analisa o jogo e faz jogadas, utilizando o Minimax
- jogada -> Armazena uma jogada, para facilitar o desenvolvimento das estruturas de dados do agente.

Relação de dependência entre os módulos:



3.2 Interface dos Módulos

3.2.1 tabuleiro

- Funções Públicas:
  - bool executaJogada(int linha, int coluna, int cor) - Coloca uma peça da cor pedida na linha e coluna passadas
  - bool jogadaValida(int linha, int coluna) - Testa se uma jogada é válida
  - int peca(int linha, int coluna) - Retorna a cor da peça presente na linha e coluna pedidas
  - int tamanho() - Retorna o tamanho do tabuleiro
  - int numBrancas() - Retorna o número de peças brancas
  - int numPretas() - Retorna o número de peças pretas

  - int num(int peca) - Retorna o número de peças da cor dada em peça
  - Tabuleiro(int tamanho) - Construtor
  - Tabuleiro() - Construtor

- Funções Privadas:
  - void init(int tamanho) - Inicializa um tabuleiro

3.2.2 tabuleiroReversi

- Funções Públicas
  - bool jogadaValida(int linha, int coluna, int cor) - Testa se uma jogada é válida
  - bool executaJogada(int linha, int coluna, int cor) - Executa uma jogada
  - int numeroJogadas(int cor) - Retorna o número de jogadas válidas de uma cor
  - int numPosicoesEstaveis(int cor) - Retorna o número de posições estáveis de uma cor
  - TabuleiroReversi(int tamanho) - Construtor
  - TabuleiroReversi() - Construtor
  - TabuleiroReversi(const TabuleiroReversi &rhs) - Construtor de Cópia

- Funções Privadas
  - bool jogada(int linha, int coluna, int cor, bool teste) - Executa ou testa uma jogada

3.2.3 tabuleiroWidget

- Funções Públicas
  - TabuleiroWidget(Placar *placar, QWidget *parent = 0, const char *name = 0) - Construtor
  - bool quemJoga() - Decide quem fará a próxima jogada

- Funções Protegidas
  - void paintEvent(QPaintEvent *) - Redesenha o tabuleiro
  - void mousePressEvent(QMouseEvent *mouse) - Atende ao pedido de um clique do mouse (executa uma jogada)

  - void mouseMoveEvent(QMouseEvent *mouse) - Atende ao pedido do movimento do mouse (verifica se uma jogada é válida)

- Funções Privadas
  - void fimJogo() - Exibe uma mensagem de fim de jogo
  - void jogada(int posX = 0, int posY = 0) - Executa uma jogada

3.2.4 agente

- Funções Públicas
  - void joga(TabuleiroReversi *tabuleiro) - Analisa o jogo e realiza uma jogada
  - Agente(int maxProfundidade, QWidget *dono) - Construtor
  - int cor() - Cor do agente
  - bool pensando() - Retorna se o agente está processando ou não
  - void mudaCor(int cor) - Muda a cor do agente
  - void mudaDificuldade(int dificuldade) - Muda o estilo de jogo do agente

- Funções Privadas
  - void run() - Inicializa a thread do agente
  - int valorMax(TabuleiroReversi *tabuleiro, int *linha, int *coluna, int alpha, int beta, int profundidade = 0) - Constroi a parte de maximização do Minimax
  - int valorMin(TabuleiroReversi *tabuleiro, int *linha, int *coluna, int alpha, int beta, int profundidade = 0) - Constroi a parte de minimização do Minimax
  - int aval(TabuleiroReversi *tabuleiro) - Avalia um dado tabuleiro
  - void sucessores(TabuleiroReversi *tabuleiro, vector<TabuleiroReversi> *sucessor, vector<Jogada> *jogada, int cor) - Retorna a lista dos possíves sucessores de um tabuleiro e as jogdas que levam a eles
  - void sucessores(TabuleiroReversi *tabuleiro, vector<TabuleiroReversi> *sucessor, int cor) - Retorna a lista dos possíveis sucessores de um tabuleiro

3.3 Jogada

- Funções Públicas
  - void colocaLinha(int linha) - Coloca a linha da jogada

- void colocaColuna(int coluna) - Coloca a coluna da jogada
- int linha() - Retorna a linha da jogada
- int coluna() - Retorna a coluna da jogada

3.4 Placar

- Funções Públicas
  - Placar(QWidget *parent = 0, const char *name = 0) - Construtor
  - void atualiza(int branco, int preto) - Atualiza o placar
  - void inicia(int branco, int preto, QPixmap *desenhoBranco, QPixmap *desenhoPreto) - Inicia o placar
- Funções Protegidas
  - void paintEvent(QPaintEvent *) - Redesenha o placar

3.5 Principal

- Funções Públicas
  - int main(int argc, char ** argv) - Inicia o programa

4 Algoritmos e Desenvolvimento

4.1 Minimax com poda Alfa-Beta

O computador joga utilizando o algoritmo Minimax. Esse algoritmo consiste, basicamente na construção de uma árvore de pesquisa com todas as jogadas possíveis e na escolha da melhor jogada possível, considerando que o adversário também jogará da melhor maneira. Como não é possível explorar toda a árvore de possibilidades, cortamos a pesquisa quando a árvore atinge uma dada profundidade e avaliamos a posição. É à partir dessa avaliação que o computador consegue escolher as melhores jogadas, portanto é muito importante que a função de avaliação seja bem escolhida. A função que utilizamos está explicada na seção Função de Avaliação.

Existem muitos estados que não valem a pena serem explorados, pois já podemos ter encontrado opções melhores de jogo. Podemos, assim, descartá-los da pesquisa, diminuindo o tempo do processamento. É para isso que serve a poda Alfa-Beta, que aplicamos junto com o Minimax.

Assim, temos o seguinte algoritmo, segundo [3]:

function BUSCA-ALFA-BETA return uma ação
input: estado
v ← VALOR-MAX(estado, -inf, +inf)

```
return a ação em SUCESSORES(estado) com valor v
function VALOR-MAX(estado, alfa, beta) return um valor de utilidade
input: estado, alfa (o valor da melhor alternativa para MAX ao longo do
caminho até o estado, beta (o valor da melhor alternativa para MIN ao
longo do caminho até o estado)
if TESTE-DE-CORTE(estado,profundidade) then
  return aval(estado)
end if
for a, s em SUCESSORES(estado) do
  v ← MAX(v,VALOR-MIN(s,alfa,beta))
  if v>=beta then
    return v
  end if
  alfa ← MAX(alfa,v)
end for
return v
function VALOR-MIN(estado, alfa, beta) return um valor de utilidade
input: estado, alfa (o valor da melhor alternativa para MAX ao longo do
caminho até o estado, beta (o valor da melhor alternativa para MIN ao
longo do caminho até o estado)
if TESTE-DE-CORTE(estado,profundidade) then
  return aval(estado)
end if
for a, s em SUCESSORES(estado) do
  v ← MIN(v,VALOR-MAX(s,alfa,beta))
  if v<=alfa then
    return v
  end if
  beta ← MIN(beta,v)
end for
return v
```

Porém, em nossa implementação, a função principal do Minimax (BUSCA-ALFA-BETA) realiza a jogada no tabuleiro, ao invés de retornar a ação.

4.2 Função de Avaliação

Uma etapa essencial na construção do algoritmo é a função de avaliação, que será utilizada para avaliar os estados durante a busca Minimax. Nesse trabalho implementamos duas funções de avaliação, permitindo ao agente dois estilos diferentes de jogo.

4.2.1 Estilo Agressivo

Nesse estilo, o agente imita um jogador iniciante de Reversi, que procura sempre maximizar suas peças e minimizar as do oponente. É um estilo fácil de vencer, mas que pode se apresentar como um oponente difícil para quem está começando a jogar Reversi.

A função pode ser representada da seguinte maneira:

$Valor = n_{minha\_cor} - n_{cor\_do\_oponente}$

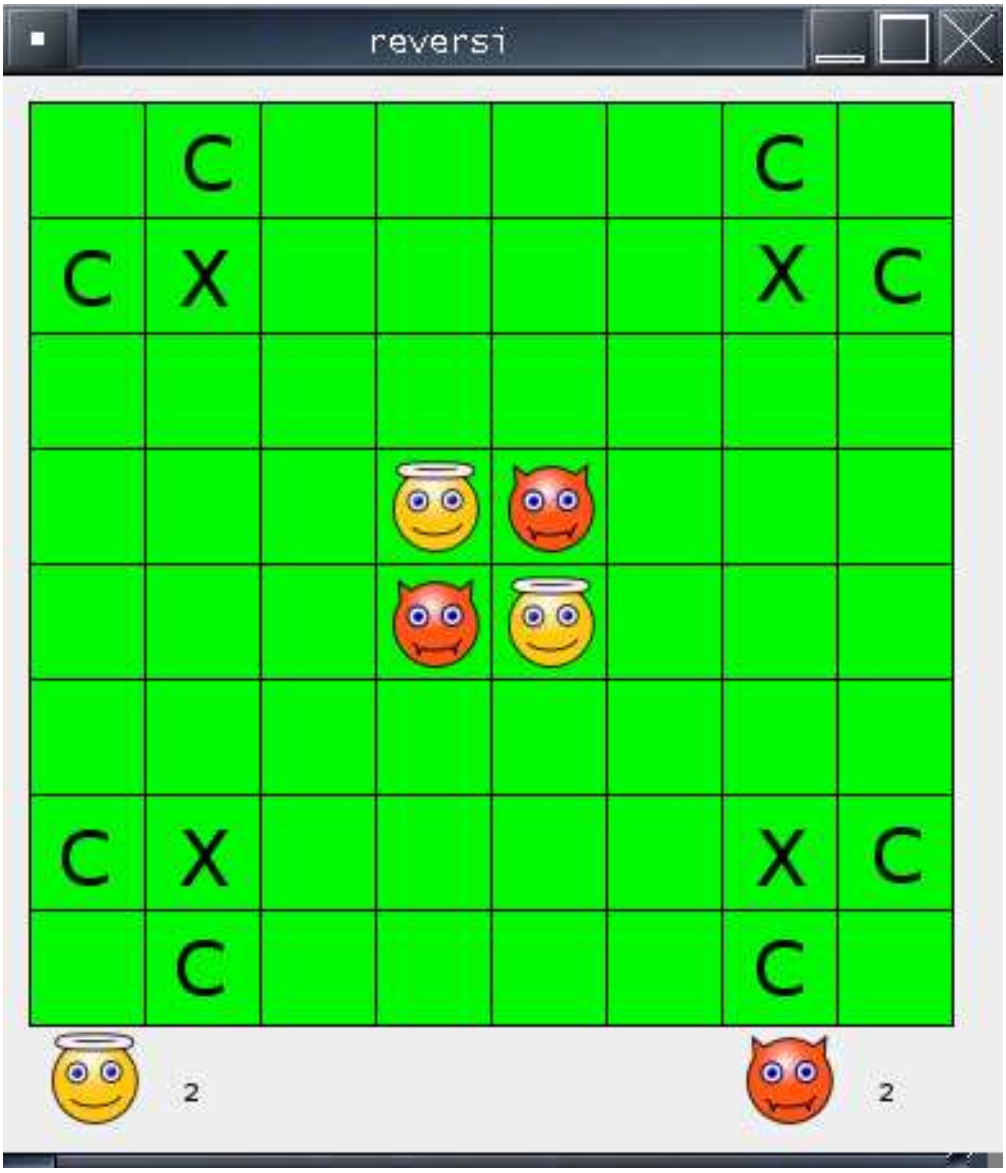
Onde n_{minha_cor} é o número de peças presentes da cor do agente e $n_{cor\_do\_oponente}$ é o número de peças do oponente do agente.

Considerando N como o número de casas no tabuleiro, esse algoritmo é O(N).

4.2.2 Estilo Estratégico

Esse estilo de jogo foi criado baseado nas instruções dadas no livro [2] Othello: A Minute to Learn, a Lifetime to Master. Ele busca os seguintes objetivos:

- Maximizar a mobilidade e minimizar a do oponente
- Evitar as casas C e as casas X, e procurar fazer com que o oponente jogue nessas casas. As casas C e X estão mostradas na imagem abaixo:



- Maximizar o número de peças estáveis e minimizar as do oponente. Peças estáveis são aquelas que não podem mais ser viradas e transformadas em peças do oponente. Peças em um dos quatro cantos do tabuleiro, por exemplo, sempre são estáveis.
- Até o meio do jogo, busca minimizar o número de peças do agente no jogo e maximizar as do oponente. No fim do jogo, busca maximizar as peças do agente e minimizar as do oponente. Essa estratégia ajuda o agente a ter uma alta mobilidade enquanto o oponente fica com poucas jogadas disponíveis.
- No início do jogo, busca dominar o centro do tabuleiro e minimizar o

domínio do oponente.

Assim, temos um jogador que busca uma alta mobilidade e procura deixar o oponente sem opções para jogar. Começa com poucas peças, dando a impressão ao oponente de ser um jogador fraco (principalmente quando o oponente é um iniciante no Reversi), mas vai buscando posições estratégicas durante o jogo, dominando o tabuleiro "aos poucos" graças a busca por posições estáveis. O oponente é surpreendido com uma rápida virada no jogo, quando o agente passa a maximizar suas peças.

A função pode ser representada da seguinte maneira:

$Valor = \sum p_i * (n_{agentei} - n_{oponentei})$

Onde p_i é o peso da característica i, $n_{agentei}$ é o valor da característica i para as peças do agente e $n_{oponentei}$ é o valor da característica i para as peças do oponente.

Onde i pode assumir os seguintes valores:

- 1 - Número de opções de jogada (Mobilidade)
- 2 - Número de peças nas casas C
- 3 - Número de peças nas casas X
- 4 - Número de peças nas casas centrais
- 5 - Número de peças estáveis
- 6 - Número de peças no tabuleiro

Para a implementação da estratégia, é importante que p_4 possua valor apenas no início do jogo, p_6 deve assumir um valor negativo até o fim do meio do jogo e positivo durante o fim do jogo e p_2 e p_3 sempre devem ter valores negativos. Os outros pesos devem ser positivos.

O número de peças estáveis é uma etapa fundamental na função de avaliação para uma boa estratégia de jogo do agente. Porém, é uma etapa cara, no pior caso percorremos todo o tabuleiro para cada posição do tabuleiro. Como é a etapa mais cara da função de avaliação, ela é $O(N^2)$.

4.3 Decisões de Projeto e Dificuldades

Buscamos fazer um projeto seguindo os princípios da Orientação à Objetos. Por isso criamos uma classe Tabuleiro e depois a classe Tabuleiro Reversi como um herdeiro. Porém, o andamento do jogo em si não é controlado por nenhuma dessas classes. A classe Tabuleiro Widget, que cria a interface gráfica do jogo, possui um Tabuleiro Reversi por agregação e controla o andamento do jogo (de quem é a vez, o jogo acabou ou não, é necessário passar a vez, etc). Essa classe também possui um Agente por agregação e o

chama sempre que sua jogada é necessária. A análise do jogo realizada pelo agente acontece em uma thread separada.

Inicialmente o jogo foi planejado como tendo uma única thread. Porém, a tela congelava após uma jogada, devido ao processamento do agente. Foi percebida então a necessidade de se utilizar uma thread separada para a análise do agente. Ao passar essa análise para outra thread, passamos por várias dificuldades devido a classe Tabuleiro Widget controlar tanto a parte gráfica quanto o processamento, o que deixou a implementação mais difícil. Seria mais fácil implementar o projeto se existisse uma thread só para a parte gráfica e outra só para o processamento do jogo.

Outra dificuldade foi o ajuste dos pesos para a função de avaliação estratégica. Ajustamos os pesos pensando em quais características do jogo seriam mais importantes e observando o resultado final. Pode ser possível, porém, ajustar ainda melhor os pesos e conseguir um agente mais difícil de ganhar.

4.4 Extras Implementados

Implementamos os seguintes recursos extras:

- Interface Gráfica Amigável - É possível jogar apenas com cliques do mouse. As opções do jogo são dadas através de caixas de diálogo. O desenho das peças é criativo e customizado.
- Dois Estilos de Jogo do Agente - É possível configurar o agente para jogar de forma agressiva, mais adequada para usuários iniciantes ou de forma estratégica, para pessoas que tem um certo conhecimento do Reversi.
- Opções de Jogo - É possível jogar contra o agente ou contra uma pessoa, com as peças brancas ou com as pretas.

5 Resultados

Para testar o nosso programa realizamos diversos jogos contra o KReversi, um programa para jogar Reversi muito usado no Linux. Usamos nos testes um limite de profundidade de sete níveis para a árvore de pesquisa. Nosso agente jogará com as pretas e o KReversi com as brancas.

No estilo agressivo contra o KReversi no nível fácil obtivemos o seguinte resultado:

1. Preto C4
2. Branco C5
3. Preto B6

4. Branco B5
5. Preto E6
6. Branco F5
7. Preto G4
8. Branco A7
9. Preto A5
10. Branco B4
11. Preto A3
12. Branco E3
13. Preto B7
14. Branco D6
15. Preto C6
16. Branco D7
17. Preto E8
18. Branco A4
19. Preto B3
20. Branco A2
21. Preto A1
22. Branco A6
23. Preto A8
24. Branco F4
25. Preto C7
26. Branco B8
27. Preto C8
28. Branco E7
29. Preto F2
30. Branco F6

31. Preto G6
32. Branco C3
33. Preto B2
34. Branco G5
35. Preto H5
36. Branco E2
37. Preto E1
38. Branco D2
39. Preto C1
40. Branco F3
41. Preto F7

Nosso agente foi vitorioso por 45 a 0. O jogo teve o seguinte resultado final:

No estilo agressivo contra o KReversi no nível médio obtivemos o seguinte jogo:

1. Preto C4
2. Branco C3
3. Preto C2
4. Branco D6
5. Preto C6
6. Branco F4

7. Preto E6
8. Branco D7
9. Preto C8
10. Branco B6
11. Preto A6
12. Branco C7
13. Preto D8
14. Branco B3
15. Preto A2
16. Branco F6
17. Preto G3
18. Branco G4
19. Preto H4
20. Branco E3
21. Preto G7
22. Branco C5
23. Preto F2
24. Branco G6
25. Preto F3
26. Branco E2
27. Preto H6
28. Branco B4
29. Preto E1
30. Branco D2
31. Preto A5
32. Branco F7
33. Preto D3

34. Branco F1
35. Preto G1
36. Branco F5
37. Preto B5
38. Branco A4
39. Preto A3
40. Branco C1
41. Preto F8
42. Branco H3
43. Preto H2
44. Branco H8
45. Preto G8
46. Branco A7
47. Preto A8
48. Branco E8
49. Preto E7
50. Branco G5
51. Preto B1
52. Branco B2
53. Preto D1
54. Branco G2
55. Preto A1
56. Branco B8
57. Preto B7
58. Preto H5
59. Branco H7

Nosso agente foi vitorioso por 41 a 23. O jogo ficou com o seguinte resultado final:

No estilo agressivo contra o KReversi no nível expert obtivemos o seguinte jogo:

1. Preto C4
2. Branco C3
3. Preto C2
4. Branco D6
5. Preto C6

6. Branco C5
7. Preto E6
8. Branco D7
9. Preto B5
10. Branco B4
11. Preto D8
12. Branco D2
13. Preto A4
14. Branco E7
15. Preto E1
16. Branco D3
17. Preto F8
18. Branco B6
19. Preto A6
20. Branco B3
21. Preto E2
22. Branco E3
23. Preto F2
24. Branco C1
25. Preto B1
26. Branco D1
27. Preto A2
28. Branco A1
29. Preto B2
30. Branco C7
31. Preto B8
32. Branco C8

33. Preto B7
34. Branco E8
35. Branco A8
36. Branco G8
37. Branco F1
38. Branco G3
39. Preto G2
40. Branco A3
41. Preto H4
42. Branco A5
43. Preto A7
44. Branco H2
45. Preto G1
46. Branco F3
47. Preto F4
48. Branco G4
49. Preto G5
50. Branco H5
51. Preto H6
52. Branco F5
53. Preto F6
54. Branco H1
55. Preto H3
56. Branco G6
57. Preto F7
58. Branco H7
59. Branco G7

Nosso agente perdeu por 59 a 4. O jogo ficou com o seguinte resultado final:

No estilo estratégico contra o KReversi no nível fácil, obtivemos o seguinte jogo:

1. Preto C4
2. Branco E3
3. Preto F6
4. Branco E6

5. Preto F5
6. Branco B3
7. Preto C5
8. Branco C6
9. Preto D6
10. Branco G5
11. Preto F4
12. Branco D3
13. Preto C3
14. Branco D7
15. Preto B4
16. Branco G6
17. Preto B5
18. Branco G3
19. Preto C7
20. Branco A5
21. Preto A6
22. Branco C2
23. Preto A4
24. Branco C8
25. Preto B6
26. Branco A3
27. Preto A2
28. Branco A7
29. Preto A8
30. Preto B2
31. Branco A1

32. Preto B7
33. Branco B8
34. Preto D8
35. Preto E8
36. Branco E7
37. Preto F8
38. Branco F7
39. Preto G8
40. Branco G7
41. Preto H7
42. Branco H8
43. Preto E2
44. Branco F3
45. Preto F2
46. Branco D2
47. Preto H6
48. Branco F1
49. Preto H5
50. Branco H4
51. Preto E1
52. Branco C1
53. Preto H3
54. Branco D1
55. Preto G4
56. Branco H2
57. Preto B1
58. Preto G2

59. Preto G1

60. Branco H1

Nosso agente ganhou por 48 a 16. O jogo ficou com o seguinte resultado final:

No estilo estratégico contra o KReversi no nível médio obtivemos o seguinte jogo:

1. Preto C4

2. Branco E3

3. Preto F6
4. Branco E6
5. Preto F5
6. Branco C6
7. Preto C5
8. Branco G6
9. Preto C7
10. Branco C3
11. Preto D3
12. Branco B4
13. Preto B5
14. Branco D2
15. Preto A3
16. Branco D6
17. Preto C2
18. Branco A4
19. Preto B3
20. Branco B6
21. Preto A5
22. Branco C8
23. Preto E1
24. Branco D1
25. Preto C1
26. Branco A2
27. Preto A1
28. Branco B1
29. Preto B2

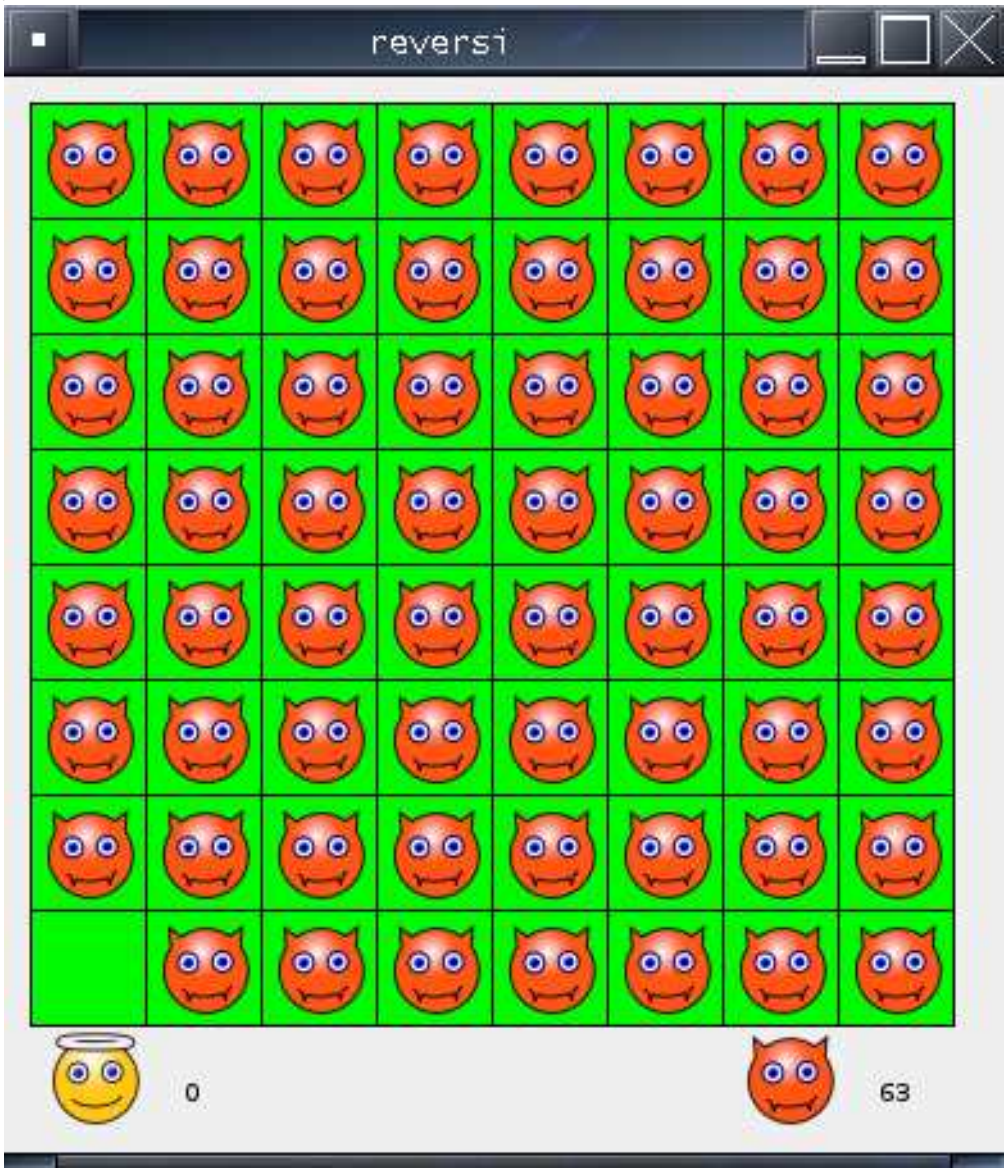
30. Branco A6
31. Preto A7
32. Branco F1
33. Preto E2
34. Branco F3
35. Preto G1
36. Preto F2
37. Preto G3
38. Branco G2
39. Preto F4
40. Branco G4
41. Preto H1
42. Preto H3
43. Branco H2
44. Preto G5
45. Branco H4
46. Preto B7
47. Preto H5
48. Preto G7
49. Branco H6
50. Preto E7
51. Branco F8
52. Preto D8
53. Branco E8
54. Preto H8
55. Branco G8
56. Preto B8

57. Preto F7

58. Branco D7

59. Preto H7

Nosso agente ganhou por 63 a 0. O jogo ficou com o seguinte resultado final:



No estilo estratégico contra o KReversi no nível Expert obtivemos o seguinte jogo:

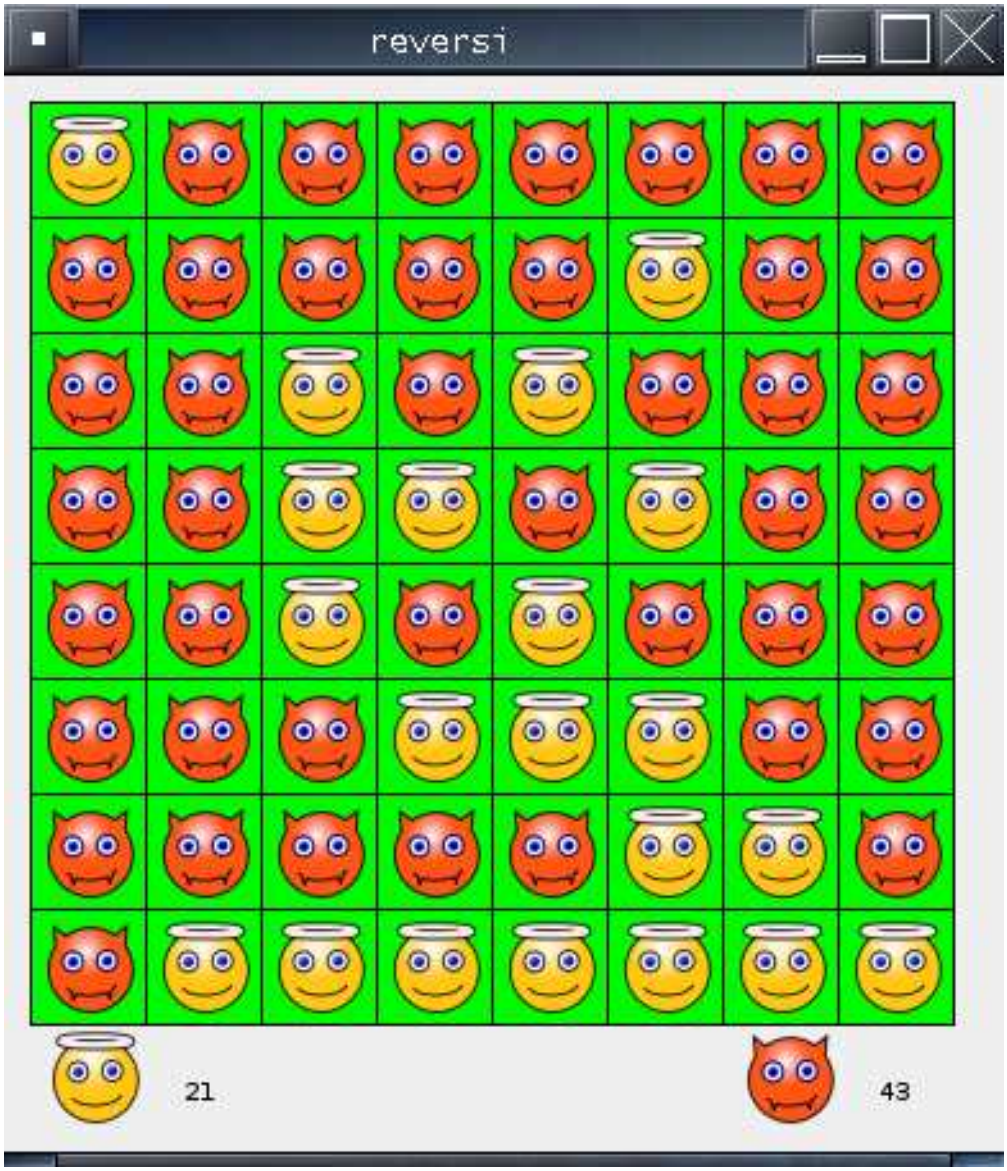
1. Preto C4

2. Branco C3
3. Preto D3
4. Branco C5
5. Preto B3
6. Branco C2
7. Preto B4
8. Branco F3
9. Preto C6
10. Branco E3
11. Preto D6
12. Branco C7
13. Preto B5
14. Branco E6
15. Preto C8
16. Branco A6
17. Preto A4
18. Branco B6
19. Preto A5
20. Branco A3
21. Preto D7
22. Branco E8
23. Preto F2
24. Branco E2
25. Preto D8
26. Branco B8
27. Preto E7
28. Branco F8

29. Preto D2
30. Branco A7
31. Preto B1
32. Branco D1
33. Preto F4
34. Branco G5
35. Preto F7
36. Branco F6
37. Preto F5
38. Branco G4
39. Preto F1
40. Branco G2
41. Preto G6
42. Branco H6
43. Preto G3
44. Branco C1
45. Preto E1
46. Branco G1
47. Preto H1
48. Branco H3
49. Preto H5
50. Branco H4
51. Preto B7
52. Branco H2
53. Preto G7
54. Branco H8
55. Preto H7

56. Branco G8
57. Preto B2
58. Branco A1
59. Preto A2
60. Preto A8

Nosso agente ganhou por 43 a 21. O jogo ficou com o seguinte resultado final:



Como pode ser visto, o algoritmo estratégico ganhou do KReversi em todos os níveis de dificuldade.

6 Conclusão

Implementamos, nesse trabalho, um agente capaz de jogar Reversi utilizando o Minimax com a poda Alfa-Beta. Fizemos dois estilos diferentes de jogo: estratégico e agressivo. Para trocar o estilo bastou trocar a função de avaliação. O estilo agressivo simplesmente busca o maior número de peças possíveis. Já o estilo estratégico utiliza estratégias conceituadas de Reversi, retiradas de [2]. Seu algoritmo é mais caro do que o agressivo, por ser mais complexo.

É interessante notar que mesmo uma função de avaliação tão simples como a do estilo agressivo permitiu um agente capaz de jogar bem, tendo vencido o KReversi no nível Fácil e Médio. A vitória no nível fácil, inclusive, foi melhor no estilo agressivo do que no estratégico. Usando o estilo agressivo ganhamos com 41 jogadas por 45 a 0. Já no estilo estratégico gastamos 60 jogadas para ganhar por 48 a 16.

Já no nível médio o estilo estratégico jogou melhor, tendo ganhado com 59 jogadas por 63 a 0, enquanto o agressivo ganhou com 59 jogadas por 41 a 23. Vemos que a vitória do estilo agressivo foi bem mais apertada. No nível expert o estilo agressivo foi incapaz de vencer, enquanto o estratégico ganhou com 60 jogadas por 43 a 21.

Vemos, assim, que para um adversário iniciante, o estilo agressivo se mostra superior ao estratégico, mesmo sendo bem mais simples. Porém, à medida em que vamos aumentando o nível do adversário, o estilo estratégico acaba sendo o mais apropriado.

Referências

[1] Artigo da Wikipedia sobre o Reversi. Pode ser acessado em http://en.wikipedia.org/wiki/Reversi

[2] Othello: A Minute to Learn - A Lifetime to Master, Brian Rose, 2005. Pode ser obtido gratuitamente em http://othellogateway.strategicviewpoints.com/rose/book.pdf

[3] Inteligência Artificial, Stuart Russel e Peter Norvig, Editora Campos, Segunda Edição.