Bruno Coswig Fiss
Kauê Soares da Silveira

Trabalho Prático

Disciplina INF01048 - Inteligência Artificial

Professor: Paulo Martins Engel

UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL

1 de julho de 2010

Sumário

Lista de Tabelas

Lista de Figuras

1 Introdução

O trabalho trata-se de desenvolver uma Inteligência Artificial para o jogo Lines of Action ([**?**]). As regras são as seguintes:

- Os jogadores alternam quem faz o movimento, sendo que as pretas fazem o primeiro lance
- As peças se movem vertical, horizontal ou diagonalmente
- Uma peça se move exatamente tantas casas quanto a quantidade de peças (aliadas ou inimigas) existentes na linha na qual está se movendo
- Uma peça pode pular por cima de peças aliadas, mas não pode pular pos cima de peças inimigas
- Uma peça pode pular em cima de uma peça inimiga (retirando a mesma do jogo), mas não pode pular em cima de uma peça amiga
- Ganha o jogador que ficar com todas as suas peças adjacentes (em qualquer direção). Caso isso seja verdadeiro para ambos os jogadores, ganha aquele que efetuou o último lance.

2 *Características Gerais do Programa*

2.1 Tecnologias

- C++
- OpenGL, GLU, GLUT, GLUI
- Linux
- Vim, Make

O programa foi escrito na linguagem C++, que alia eficiência e acesso aos recursos de baixo nível da máquina com orientação a objetos e suporte à abstração.

A interface gráfica utiliza-se da biblioteca GLUI ([**?**]), a qual está no topo de uma cadeia de bibliotecas (OpenGL, GLU, GLUT, GLUI) em cuja base se encontra OpenGL, sendo que todas são livres e portáveis.

O desenvolvimento ocorreu no sistema operacional Linux, utilizando ferramentas como Vim e Make. Como a portabilidade foi um dos enfoques, ao final também foi gerada, sem grandes dificuldades, uma versão para Windows.

2.2 Estruturas de Dados

- Bitvector

Foi utilizada a classe bitvector da biblioteca padrão da linguagem ([**?**]). As peças são representadas em um vetor de 64 posições contendo o valor 1 nas casas ocupadas e 0 nas casas vazias. O tabuleiro é composto por dois destes vetores, um para cada jogador.

Para determinar casas ocupadas, independente da cor, utilizamos um terceirovetor que é o resultado do ou lógico, efetuado bit a bit, dos outros dois.

Para determinar os movimentos válidos, utilizamos vetores pré-computados com a quantidade de peças em cada linha, em cada coluna, em cada diagonal principal e em cada diagonal secundária.

Para detectar que todas as peças de um jogador estão adjacentes é utilizado um algoritmo recursivo, semelhante ao floodfill, que também utiliza um bitvetor, desta vez para marcar quais casas já foram visitadas.

O tabuleiro também armazena uma cache dos valores da heurística (ver seção **??**).

2.3 Algoritmo

- Negamax
- Poda alphabeta
- Best choice first
- Cache dos valores das heurísticas

Utilizamos o algoritmo negamax [**?**], em que cada jogador objetiva maximizar o simétrico da resposta ótima do adversário. Acrescentamos a poda alpha beta [**?**], na qual variações que comprovadamente não levarão a um valor melhor de heurística não são avaliadas.

A cada iteração, o valor da heurística é calculado em todos os sucessores do nó atual e estes são ordenados por este valor, de forma que expandimos primeiro os nós que têm mais chance de se mostrarem melhores e causarem uma poda na árvore.

Também foi implementada uma cache dos de heurística (detalhados na seção **??**), de tal forma que apenas os valores do jogador que acabou de jogar são recalculados, sendo os valores referentes ao outro jogador aproveitados da iteração anterior.

2.3.1 Heurística

- Fator Aleatório
- Distância para o centro
- Concentração
- Centralização
- Centro de Massa
- Quadrados
- Conectividade
- Uniformidade

Foram implementadas 7 funções heurísticas, inspiradas em [**?**]. Cada uma delas calcula o valor correspondente a cada um dos jogadores e retorna a diferença (normalizada entre 0 e 1) entre estes valores.

Algumas utilizam o valor do centro de massa, calculado como a média das coordenadas das peças do jogador em questão.

Distância para o centro

É o inverso da média das distâncias euclidianas das peças para o centro. Objetiva recompensar configurações em que temos várias peças no centro, pois é necessário passar pelo centro para reunir as peças, e se for possível reuní-las no centro é melhor ainda.

Concentração

É o inverso da média das distâncias de manhatan das peças para o centro de massa. O centro de massa representa um bom ponto para tentar concentrar as peças, sendo uma alternativa ao centro absoluto (que é o caso da heurística anterior). Para não privilegiar configurações com menos peças, subtraímos uma constante pré-computada que representa o valor mínimo possível de ser assumido pela heurística dada a quantidade de peças atual (por exemplo, com 3 peças este valor é um, atingido quando uma das peças está sobre o centro de massa e as outras duas estão adjacentes a esta).

Centralização

É a média da qualidade das posições das peças, segundo a seguinte tabela de pesos:

-80	-25	-20	-20	-20	-20	-25	-80
-25	10	10	10	10	10	10	-25
-20	10	25	25	25	25	10	-20
-20	10	25	50	50	25	10	-20
-20	10	25	50	50	25	10	-20
-20	10	25	25	25	25	10	-20
-25	10	10	10	10	10	10	-25
-80	-25	-20	-20	-20	-20	-25	-80

Tabela 2.1: Tabela de Pesos da Heurística de Centralização

O objetivo é penalizar peças na borda do tabuleiro (que são facilmente bloqueáveis pelo inimigo e que têm mais chance de estar longe do grupo principal) e recompensar as peças que estão mais perto do centro.

Centro de Massa

É a distância euclidiana do centro de massa para o centro do tabuleiro. Se o centro de massa está no centro do tabuleiro, as peças podem estar arbitrariamente longe (como no início do jogo, por exemplo). Quanto mais longe do centro do tabuleiro estiver o centro de massa, mais próximas as peças têm que estar.

Quadrados

É a quantidade de quadrados de lado 2 com pelo menos três peças do jogador em questão e a no máximo duas casas de distância do centro de massa. A vantagem de se ter estruturas deste tipo é que são necessário pelo menos dois movimentos para desconectá-las. A razão para exigir uma distância mínima do centro de massa é evitar que se formem grupos isolados.

Concectividade

É o número médio de peças aliadas adjacentes das peças do jogador em questão. Objetiva recompensar aglomerações de peças em oposição a formação em linha, em que uma peça só está conectada com outras duas, as quais são mais fáceis de destruir.

Uniformidade

É o inverso da área do menor quadrado de lados paralelos aos lados do tabuleiro que engloba todas as peças do jogador em questão. Objetiva recompensar configurações com peças em linhas e colunas próximas.

2.3.2 Algoritmo Genético

- Algoritmo Genético
- ...

O valor final da função de avaliação foi definido como sendo uma média ponderada das heurísticas. Para determinar a melhor tupla de valores de pesos, utilizamos um algoritmo genético, detalhado a seguir.

Representação do Problema

A representação escolhida foi uma tupla com os pesos (entre 0 e 1) dados para cada função heurística.

Geração da População Inicial

A população inicial (parâmetro) é gerada sorteando-se valores de pesos aleatórios para cada heurística de cada indivíduo.

Avaliação da População

A avaliação é feita simulando partidos de todos contra-todos em dois turnos (alternando quem faz a jogada inicial) e somando-se a quantidade de vitórias obtidas.

Seleção, Recombinação e Mutação

Após feita a avaliação de uma geração, um certa porcentagem (parâmetro) dos indivíduos mais adaptados é escolhida para permanecer (clonagem). Então os indivíduos restantes da população são fruto da recombinação destes. Essa recombinação se dá primeiro escolhendo aleatoriamente dois indivíduos diferentes para serem combinados, e então aleatoriamente de qual destes indivíduos cada peso é herdado. Ainda há uma certa porcentagem de chance (parâmetro) de um destes pesos ser sortiado aleatoriamente novamente, caracterizando uma mutação.

Critério de Parada

O algoritmo para após uma certa quantidade de iterações (parâmetro).

Fator Aleatório na Função Heurística

Para evitar que houvessem empates por repetição de jogadas, foi incluído um fator aleatório na heurística, variando entre 0 e 1, cujo peso também foi incluído na composição dos indivíduos.

Parâmetros Utilizados

O algoritmo foi executado com uma população de 10 indivíduos, uma porcentagem de seleção de 50

Heurística	Peso
Fator Aleatório	0.00763
Distância para o centro	0.38113
Concentração	0.63358
Centralização	0.89597
Centro de Massa	0.03325
Quadrados	0.41463
Conectividade	0.78261
Uniformidade	0.01670

Tabela 2.2: Resultado da Primeira Execução do Algoritmo Genético

Pode-se verificar que o peso do fator aleatório ficou quase 0, como era de se esperar. Além disso, outras duas funções heurísticas ficaram com peso bem baixo (Centro de Massa e Uniformidade). Dada a complexidade do cálculo destes funções, decidimos eliminá-las da heurística para poder atingir uma profundidade maior. Executamos o algoritmo genético com os mesmos parâmetros, porém eliminando estas funções heurísticas (tabela **??**).

Heurística	Peso
Fator Aleatório	0.00616
Distância para o centro	0.87480
Concentração	0.07819
Centralização	0.66735
Quadrados	0.41734
Conectividade	0.32783

Tabela 2.3: Resultado da Segunda Execução do Algoritmo Genético

Desta vez obtivemos um peso baixo para a função heurística Conectividade. Pelas mesmas razões citadas anteriormente, executamos novamente o algoritmo removendo mais esta função (tabela **??**). Finalmente atingimos pesos satisfatórios, ou seja, todas as heurísticas têm uma contribuição significativa para a avaliação final.

Heurística	Peso
Fator Aleatório	0.00251
Distância para o centro	0.82723
Concentração	0.97915
Centralização	0.96032
Quadrados	0.96950

Tabela 2.4: Resultado da Terceira Execução do Algoritmo Genético

2.3.3 Resultados

- Profundidade

A versão final do algoritmo, dado o tempo limite de 5 segundos para efetuar cada lance, alcança 5 ply.