Inteligência Artificial

Y@IST

Grupo 26
Daniel Cardoso, nº 66964
Francisco Raposo, nº 66986
Miguel Aragão, nº 67043

LEIC-A
3º Ano, 1º Semestre
2011 – 2012

O projecto teve 3 fases distintas de desenvolvimento, em todas elas surgiram problemas, que vamos tentar explicar de uma forma sintetizada, referindo quais as soluções adoptadas, e de que forma chegámos a essas conclusões.

1ª Fase

Numa primeira fase, desenvolvemos as funções relativas à manipulação do tabuleiro e não surgiram grandes complicações, a não ser dúvidas acerca da sintaxe do lisp, uma vez que foi o primeiro contacto com um projecto realizado nesta linguagem. Não existem portanto grandes considerações, a apresentar acerca desta primeira etapa.

2ª Fase

Numa segunda fase, preocupámo-nos com a interface, com a função relativa ao jogador manual e finalmente a função que determina se um tabuleiro representa uma situação de jogo final, ou seja, determina se está perante uma jogada que dá a vitória ao jogador que jogou em último. Foi nesta altura que surgiram, as primeiras dificuldades em termos de algoritmo e que passamos a explicar.

Ao depararmo-mos com a necessidade de saber se um caminho é uma situação final, tivemos de desenvolver um algoritmo que de uma forma eficaz, decidi-se o que é uma situação válida de vitória. Para isso, implementámos duas funções, uma que verifica se existem peças iguais no anel exterior e em cada terço do tabuleiro, uma vez que para um caminho ser considerado final, é necessário ter pelo menos uma peça nessas condições, e a outra função, que apenas é chamada quando a primeira devolve true, que determina se existe um caminho válido.

A primeira função não nos trouxe grandes problemas uma vez que se trata de uma simples verificação das posições ocupadas no tabuleiro, para determinar se existem peças ou não. A dificuldade maior surgiu quando começámos a desenvolver o algoritmo da segunda função pois precisámos de pensar numa forma de percorrer o tabuleiro respeitando as regras impostas pelo enunciado relativas à adjacência entre posições.

De uma forma resumida, e uma vez que um caminho final terá de obrigatoriamente passar por posições do anel exterior do tabuleiro, optámos por ir verificar cada posição desse mesmo anel, e ir gerando o caminho através das peças adjacentes à posição actual, de forma a poder verificar todas as possibilidades presentes no tabuleiro, e assim garantir que são detectadas correctamente as situações de vitória dos jogadores. Assim que é detectada uma situção final, o jogo termina portanto, sendo vencedor, o jogador que jogou em último lugar.

Conseguimos assim, concluir esta 2ª fase do desenvolvimento do projecto, não tendo

surgido grandes complicações, sendo de realçar então, como maior barreira até este momento, a implementação da função anteriormente descrita.

3ª Fase

Como 3ª e última fase, e como vamos explicar mais à frente, temos portanto a conclusão do projecto, passando pela implementação do jogador automático inteligente, que envolveu uma maior dedicação da nossa parte quer em termos de tempo gasto a programar, quer em aplicação dos métodos leccionados durante o semestre, sendo também a parte em que de certa forma vimos concretizados os algoritmos apresentados pelo professor, e que já tinham sido estudados por nós principalmente a nível teórico.

Podemos dividir, esta 3ª e última fase, em 3 partes distintas, que representam as grandes dificuldades encontradas, passando a explicar cada uma delas, expondo também as nossas soluções aos problemas. Começando por nos preocuparmos com a forma como iríamos implementar e utilizar o algoritmo min max, chegámos à altura em que sentimos necessidade de ter um limite de tempo de procura, e ainda, e por fim, deparámo-nos com as dificuldades inerentes ao desenvolvimento de uma função heurística.

Algoritmo Min Max

Para a realização desta parte do projecto, tivemos de efectuar algum trabalho de pesquisa uma vez que embora tenha sido matéria de estudo durante as aulas, sentimos a necessidade de obtermos mais material acerca deste assunto, pois para o correcto funcionamento do nosso jogador automático, teríamos de ter implementado um algoritmo min max sem falhas e eficaz.

Decidimos, numa fase inicial, centrar as nossas atenções em apenas compreender a fundo o objectivo deste método de pesquisa, de forma a criar de uma forma clara, e sem dúvidas uma implementação capaz de comunicar com todos os elementos do projecto, conseguindo assim uma base sólida para o produto final. Concluímos, após um cuidado estudo, que iríamos implementar um algoritmo min max com cortes alpha e beta, pois foi a forma mais eficaz que encontrámos, e tinha já sido utilizado por nós nos exercícos das aulas práticas, e do próprio teste teórico.

Tendo terminado a implementação do algoritmo, decidimos, de forma a testar o correcto funcionamento deste, e uma vez que não tínhamos ainda criado uma função heurística que determina a “qualidade” da jogada, atribuímos valores apenas à derrota (-1), ao empate (0), à situção a meio de jogo (1) e à vitória (2). Assim concluímos o desenvolvimento do min max, que nos garantia já uma boa base para a inteligência do nosso jogador, uma vez que seria agora necessário “apenas” criar uma função heurística adequada.

Limite de Tempo

Tendo terminado a implementação do algoritmo min max, tínhamos consciência que, embora estivesse correcta a sua codificção, teríamos um problema, pois a sua aplicação nos tabuleiros do jogo iria implicar um enorme recurso de tempo, pois a quantidade de opções cresce exponencialmente com o tamanho dos mesmos, ou seja, decidimos encontrar uma solução antes de querer criar um heurística válida, pois assim teríamos, finalmente tudo pronto, de modo a poder focar toda a nossa atenção na inteligência em si.

Tendo discutido entre nós e com outros grupos de colegas, decidimos utilizar um método de procura em profundidade limitada, sendo que o que estava até ao momento implementado era o método de procura em profundidade, e que demorava muito tempo a encontrar uma solução válida. Colocando essa "tarefa" na função relativa ao jogador automático, criámos assim um método que nos pareceu o mais eficaz, de obter uma jogada proveniente do algoritmo min max, tendo controlo sobre o tempo que a máquina dispendia na sua execução.

Foi nesta altura que surgiu a noção da importância do argumento tempo da função *executa-jogo* descrita no enunciado, pois após a implementação do método de procura em profundidade limitada, pensámos, qual será a profundidade limite a que o algoritmo deverá chegar? Será imposta por nós?

A partir deste momento, achámos fundamental, controlar o tempo que o jogador automático deveria ter para "pensar", e assim definir a profundidade limite em função do tempo imposto na invocação do jogo. Decidimos assim que um jogador automático deveria ir executando o algoritmo min max com um certo limite de profundidade, e, caso tivesse ainda tempo disponível, executar o mesmo algoritmo mas agora a um nível de profundidade superior. Acabámos assim por implementar o método de procura em profundidade limitada iterativo, em que o jogador enquanto tem tempo, chama o algoritmo min max com profundidade 1, depois 2, depois 3...e assim sucessivamente.

No final obtivemos, pensamos nós, um método eficaz de procura, que nos garante uma jogada com um certo nível de inteligência, dependendo portanto da melhor ou pior definição, da função heurística.

Função Heurística

Chegando a esta parte final, mas muito provavelmente a mais importante do projecto, sendo que em conjunto com os outros pontos desta terceira fase definem o nível de inteligência com que o nosso jogador irá definir por que jogada optar, decidimos à semelhança da implementação do algoritmo min max, primeiramente preocuparmo-nos apenas em pensar, numa boa solução sem começar a implementar possíveis ideias fracas,

ou mesmo erradas.

Tendo finalmente decidido a forma como queríamos que o nosso jogador pensasse, a forma como queríamos que ele jogasse, podemos separar dois pontos fundamentais da nossa heurística final, os pontos associados ao algoritmo que analisa caminhos e os pontos de bónus associados a certas situações específicas de que nos fomos lembrando.

Podemos já referir, que, todos os valores que forem apresentados de seguida terão em conta que ao caso de vitória são atribuidos 19000 pontos, para o caso de empate 0 pontos, e, para o caso de derrota são atribuídos -19000 pontos, sendo todos os valores referentes a outras situações situados entre estes dois (-19000, 19000).

Passamos a explicar, de uma forma que esperemos que seja perceptível, uma vez que por vezes não é muito fácil compreender a forma de pensar, de outras pessoas mesmo sendo na nossa cabeça evidente.

– **Pontos associados aos caminhos:**

Numa primeira fase, não nos preocupámos com situações específicas de jogo possíveis, mas sim com o facto de ser importante a construção de caminhos válidos, ou seja a construção de caminhos que obedecem às regras de adjacência impostas no enunciado. Surgiu essa ideia na nossa cabeça uma vez que já tínhamos relativa facilidade na manipulação deste tipo de problema, pois durante a implementação da função que verifica se um caminho é final, tivemos de dominar na perfeicção estes problemas. Assim sendo, e com apenas ligeiras diferenças em relação ao algoritmo referido na última frase, optámos por disponibilizar 7000 pontos, que são distribuídos de forma equilibrada pelas várias transições que se verificam durante a definição do caminho. De forma a manter esses valores dentro do limite, e considerando que no máximo poderá haver um limite de transições para posições novas (por novas entende-se um posição que ainda não foi gerada e colocada na lista), de *((nº total de posições no tabuleiro / 2) +1)*, divide-se assim os tais 7000 pontos, garantindo que não são utrapassados.

Uma vez implementado correctamente este procedimento, coube-nos apenas, e mais do que uma vez, dada a necessidade de ir acertando os valores para outros que nos foram parecendo mais correctos, determinar a percentagem de pontos possíveis para cada transição que iriam ser adicionados ao valor da heurística, por exemplo uma transição que recebe 90% do valor máximo para uma transição, é considerada mais importante, mais “inteligente” que uma que apenas recebe 40 %.

Concluímos assim, a nossa explicação acerca desta parte da atribuição de pontos à heurística, esperando que tenha sido entendida pelo leitor, uma vez que de uma forma resumida, pensamos ter referido o essencial sobre a mesma.

– **Pontos de bónus:**

Tendo testado contra jogadores de outros grupos, e contra nós mesmos, fomos sentindo a necessidade de dar importância a situações de bónus que íamos considerando importantes ao longo dos testes, ou porque perdíamos ou apenas porque achávamos vantajosas para um possível adversário diferente.

Vamos agora referir e resumir cada uma dessas situações que têm como objectivo tornar mais forte a nossa estratégia de jogo.

Enquanto estamos a analisar o caminho, dentro do ciclo que vai gerando as transições todas a que damos valores diferentes da heurística, temos em conta um dos bónus que se baseia, no facto de acharmos importante ter peças em vários terços diferentes e ainda se essas peças estão no anel exterior ou não, dando mais pontos neste último caso. Temos em atenção o facto de apenas podermos somar este bónus uma vez de forma a não ter problemas com o valor máximo possível para a heurística, uma vez que a vitória são 19000 pontos.

Após determinarmos o valor pontual referente ao caminho, temos uma verificação que nos indica quantas adjacências tem a peça com mais adjacências nesse caminho, e assim damos pontos de bónus a jogadas que colocam peças em locais com bastantes possibilidades de escape.

No final, antes de devolver a heurística ao algortimo min max, verificamos se o tabuleiro analisado possui peças em todos os terços no anel exterior, e damos um considerável bónus a essa situação, uma vez que achamos importante garantir peças nessas localizações.

Por fim, ainda verificamos uma situação que considerámos importante, que decide atribuir, um grande bónus a jogadas que colocam peças em vários anéis diferentes uma vez que a movimentação entre anéis permite uma escolha de caminhos diversos em busca da vitória bastante maior, pois faz com que o adversário tenha de andar atrás de nós para nos bloquear as investidas por vários sítios, ficando assim vulnerável noutras posições.

Acabamos assim por resumir todos os principais factores na atribuição de pontos elaborada pela nossa função heurística, função essa que considerámos eficaz ao longo dos jogos efectuados tanto contra outros humanos tanto contra outros jogadores automáticos.

Conclusão

Pensamos que nos serviu de grande exercício mental a execução deste projecto, e definitivamente motivou-nos o estudo dos algoritmos, e das matérias leccionadas durante as aulas, uma vez que com este projecto, tivemos obrigatoriamente de utilizar técnicas e métodos que fomos estudando durante o semestre e que em algumas situações, não estavam totalmente entendidos, uma vez que a nível teórico escapam algumas considerações que se tornam claras, durante a aplicação dos conceitos, e vice-versa.

Concluímos assim o nosso relatório, contentes com o nosso produto final, e cientes de que nos esforçámos e tentámos produzir um interessante e inteligente jogador automático, que foi claramente o grande desafio deste trabalho, como era objectivo.