Instituto Tecnológico de Aeronáutica

Divisão de Ciência da Computação
Departamento de Teoria da Computação
CTC-15: Inteligência Artificial

Trabalho de Exame

Planet Wars

Alunos
Cesar R. Kawakami
Guilherme R. N. Souza
Leonardo Ribeiro Carvalho
Rodrigo Simões Almeida

Professor
Carlos Henrique Ribeiro

02 de Dezembro de 2010

1 Objetivos

Fazer um jogador autônomo para Planet Wars, jogo usado na competição do Google Ai-Contest do ano de 2010. Esse jogador deve aplicar os conceitos e algoritmos vistos na matéria de Inteligência Artificial.

2 Introdução

Planet Wars (Figura 1) é um jogo entre dois ou mais jogadores, cujo objetivo é, ao final dos turnos, ter a maior frota de naves dentre todos os participantes. Apesar poder ser jogado com vários jogadores, todas as partidas realizadas, tanto nos testes como na competição, foram realizadas entre dois adversários.

O jogo é dividido em turnos no qual ambos jogadores tem todas informações relevantes disponíveis para tomar suas decisões e enviar frotas para combate. O combate entre frotas ocorre nos planetas, cada nave destrói exatamente uma nave inimiga e também é destruída. Ao final do combate quem tiver o maior número de naves sobreviventes no planeta o domina. Se

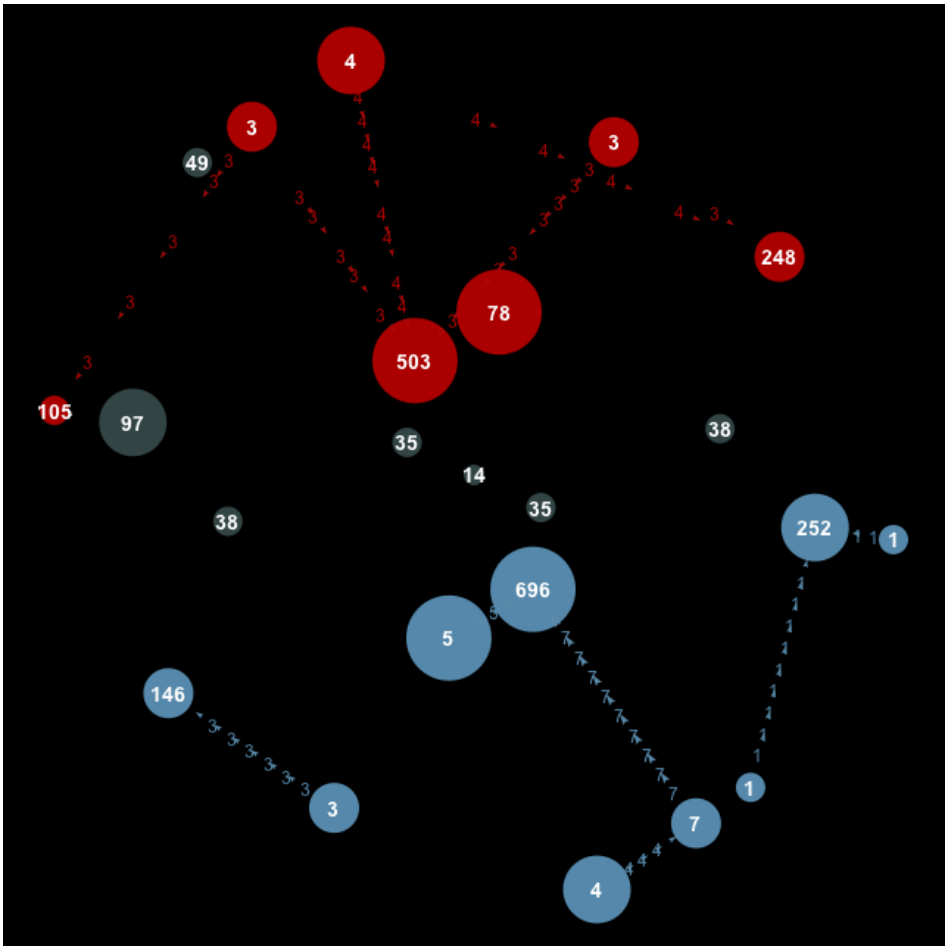

Figura 1: Planet Wars

restarem 0 naves o dono do planeta não muda. O jogador com planetas sob seu domínio tem sua frota local aumentada de acordo com a taxa de crescimento específica, a qual depende do tamanho do planeta dominado.

A abordagem para construção do jogador autônomo (a partir de agora denominado *bot*) dividiu-se em duas partes. A primeira foi o desenvolvimento de um *bot* que sabe se comunicar com o juíz/servidor e implementa heurísticas (estratégias básicas relacionados ao entendimento do funcionamento geral do jogo). Exemplos de fatores levados em consideração na heurística são: tentar defender suas naves, destruir naves inimigas, benefício trazido por conquistar planetas, entre outros. Mais informações na seção 3.

Essa versão inicial foi colocada no servidor do Google Ai-Contest e já apresentou bons resultados. A heurística, contudo, dependia de um conjunto de trinta e dois pesos associados a diferentes situações de jogo. Esses pesos haviam sido escolhidos sem muito critério, apenas seguiram um lógica geral relacionada ao significado de cada peso. Para conseguir um melhor conjunto de pesos foi utilizado um algoritmo genético de evolução também detalhado na seção 3.

3 Implementação

3.1 *Bot* com estratégia básica

Inicialmete foi implementado um *bot* com estratégia básica. Toda a avaliação do estado do jogo considera um certo número de jogadas a frente, chamado horizonte. Esse horizonte funciona como uma poda de considerações muito distantes que poderiam atrasar o jogador e evita considerações que não fazem sentido, pois a previsão não considera que haverão jogadas, i.e., essa aboradagem avalia o jogo daqui a um número de jogadas se todos os jogadores tivessem como suas próximas ações não fazer nada. Esse horizonte é calculado levando em consideração a distância média entre planetas pertencentes ao *bot* e os planetas dominados pelo adversário.

A estratégia de jogo pode ser definida, de maneira geral, pelos passos:

1. Para cada planeta acha-se quantas naves podem ser mandadas para o ataque sem que o planeta fique muito desprotegido e possa ser facilmente conquistado. Para isso, analisa-se as frotas que o estão atacando, achando quantas naves de defesa ele perderia a cada turno até o horizonte. O valor restante de naves ao atingir o horizonte é o valor que é considerado disponível para ataque e defesa, um valor negativo significa perda do domínio.

2. Cada planeta com naves de crédito tenta defender outros planetas. Cada planeta pertencente ao jogador busca outros planetas que serão consquistados pelo inimigo (previsão negativa no item anterior) tentando fornecer naves suficientes para evitar essa perda. Essa ajuda a planetas vizinhos está condicionada à manutenção do domínio do planeta de origem das naves, caso a ajuda signifique perder o planeta original a ajuda não acontecerá. A ajuda também considera o tempo que a ajuda demorará para chegar ao planeta ajudado, se ele for conquistado antes de ser possível chegar a ajuda ela não será enviada.

3. Com as naves restantes, após defender a si mesmo e fornecer ajuda a planetas aliados, cada planeta busca entre os planetas neutros e do inimigo aquele que apresenta o maior ganho esperado e realiza o ataque. Para cada planeta, é calculado um valor que representa o lucro esperado por atacá-lo. Dentro desse lucro é levado em consideração naves perdidas no ataque, naves perdidas pelo inimigo, quantas naves seriam ganhas e quantas naves o inimigo deixaria de ganhar devido à taxa de crescimento fornecida pelo planeta. Cada um desses critérios é ponderado com o uso de quatro pesos, que representam uma parte do estado do jogo tais como a distância do planeta aos outros (amigos e inimigos) o relação do lucro de cada jogador e a relação de naves totais de cada jogador.

Essa estratégia considera alguns dos pontos tidos como essenciais para o bom desempenho do jogador e leva em consideração diversos jogos assistidos. A partir desta versão é possível fazer variantes jogarem entre si e selecionar a melhor dentre as opções geradas.

3.1.1 Detalhamento em algoritmo da estratégia utilizada

A estratégia de jogo, em cada turno, é dada por

procedimento FaçaTurno()
para todo planeta meu p **faça**
  DefendeOutrosPlanetas(p)
  AtacaOutroPlaneta(p)
fim para
fim procedimento

procedimento DefendeOutrosPlanetas(planeta meu p)
para todo planeta meu q **faça**
  se existe $1 \leq t \leq$ MAX_TURN com Previsão(q, t) < 0 **então**
    $n \leftarrow -$Previsão(q, t)
    se naves sobrando em $p > n$ **e** Distância(p, q) $< t$ **então**
      EnviarFrota(p, q, n)
    fim se
  fim se
fim para
fim procedimento

procedimento AtacaOutroPlaneta(planeta meu p)
$horizonte \leftarrow \frac{3}{2}$Média($\{$Distância$(x, y) : x \in$ meus planetas$\wedge y \in$ não-meus planetas$\}$)$+10$
para todo planeta não-meu q **faça**
  se Distância(p, q) $\leq horizonte$ **então**
    $residuo \leftarrow -$Previsão(q, $horizonte$)
    se $residuo \geq 0$ **e** $residuo \leq$ naves sobrando em p **então**
      {se o planeta não for meu e for possível tomar}
      $d \leftarrow$ Distância(p, q)
      $ganhonaves \leftarrow (horizonte - d) \cdot$ TaxaDeCrescimento(q)
      $ganhototal \leftarrow (peso_0 + peso_1) \cdot residuo + (peso_2 + peso_3) \cdot ganhonaves$
    fim se
  fim se

13: **fim para**
14: EnviarFrota(p, q com melhor $ganhototal$, $residuo$ correspondente)
15: **fim procedimento**

A função Previsão(q, t) simula, dadas as naves atualmente no espaço, todas as chegadas de frota até o tempo t, e retorna Previsão$(q,t) > 0$ o número de naves minhas no planeta se for meu e Previsão$(q,t) < 0$ o número de naves oponentes no planeta se não for meu. Além disso, $peso_i$ obedece à expressão

$$\begin{aligned} peso_i = {} & param_{k,i,0} \\ & + param_{k,i,1} \cdot \frac{\text{Média}(\{\text{Distância}(q,x) : x \in \text{planetas inimigos}\})}{\text{Média}(\{\text{Distância}(q,x) : x \in \text{planetas}\})} \\ & + param_{k,i,2} \cdot \frac{\sum_{\text{planeta inimigos } j} \text{TaxaDeCrescimento}(j)}{\sum_{\text{planeta } j} \text{TaxaDeCrescimento}(j)} \\ & + param_{k,i,3} \cdot \frac{\text{TotalNavesInimigas}}{\text{TotalNaves}} \end{aligned}$$

em que $0 \leq k < 2$ varia se o planeta sendo considerado é neutro ou inimigo, e $param_{k,i,j}$ são os parâmetros de ponderação que serão treinados por meio de evolução por algoritmo genético.

3.2 Algoritmo genético

Para a execução do algoritmo foi definido como cromossomo o conjunto de trinta e dois pesos ($param_{i,j}, 0 \leq k < 2, 0 \leq i < 4, 0 \leq j < 4$), sendo cada peso um alelo. Uma população é o conjunto de sessenta cromossomos. A população inicial foi gerada aleatoriamente.

Após gerar a população inicial, o programa entra em um loop infinito de treinamento no qual repete as seguintes ações: salva a geração, promove a seleção natural e obtém a próxima geração. Salvar a geração significa simplesmente salvar os pesos de todos os elementos que a compõe em um arquivo. Esses arquivos funcionam como uma espécie de controle de versão.

A seleção natural dos indivíduos e a obtenção da próxima geração são realizadas através do processo:

- Seleciona, dentro dos sessenta elementos da população, seis jogadores que irão participar de um campeonato
- Realiza o campeonato, que consiste fazer todas as combinações dois a dois desses jogadores e para cada dupla realizar jogos em 10 mapas aleatórios dentre os 100 existentes
- Os jogadores são ordenados por número de vitórias (contando uma por mapa) e a seleção de quem produzirá descendentes segue o procedimento conhecido como *Tournment Selection* com probalidade de escolher o campeão de 70%
- Os passos anteriores são repetidos até terem sido escolhidos 20 "campeões"
- É produzida a nova geração. Dado o conjunto de indivíduos que produzirão descendentes são escolhidos dois aleatoriamente e é feito um crossover uniforme gerando dois indivíduos. Isso é feito sessenta vezes, até gerar duas vezes o tamanho da população.

- Esses indivíduos sofrem mutação com probabilidade de 4%
- Desses cem indivíduos são escolhidos sessenta aleatoriamente que comporão a próxima geração

O programa foi feito de modo que uma geração pode ser evoluída de maneira paralela, dividindo jogos entre diversos computadores. Essa funcionalidade, junto às técnicas probabilísticas de efetuar campeonatos e seleção de população, foi muito importante, já que a evolução é um gargalo na obtanção de um bom *bot*, justamente por ser um processo demorado.

4 Resultados

O *bot* que participou da competição oficial no Google Ai-Contest foi a versão original, sem evolução, obtendo a posição 524 de um total de 4617 participantes. O resultado obtido pode ser considerado satisfatório frente ao alto nível dos competidores.

O processo de evolução foi realizado duas vezes. Na primeira o *bot* original fez parte da primeira população e o resultado foi um bot praticamente equivalente ao primeiro considerando os resultados, apesar do valor dos pesos serem diferentes. A convergência nesse caso demorou somente 25 turnos o que gerou grande suspeita de um máximo local. Alguns parâmetros (geração da população original, tamanho da população, entre outros) foram levemente alterados na tentativa de obter um novo conjunto de pesos melhor.

No decocorrer da segunda evolução, mais especificamente na geração 44, foi possível notar que a população havia mais uma vez convergido, a maior parte dos indíviduos eram razoavelmente semelhantes. Foi realizado um teste, jogando todos os mapas, realizando o confronto da versão original do *bot* contra a versão evoluída. O placar obtido foi 56 vitórias da nova versão, 43 vitórias da versão original e 1 empate. Apesar de não ser uma vitória expressiva, mostra evolução.

No torneio realizado em sala de aula, o *bot* resultante obteve a segunda colocação, perdendo apenas uma partida.

5 Possíveis melhorias

O programa poderia levar em conta outros fatores do estado do jogo para a avaliação do lucro de um planeta. Uma estratégia não implementada que é utilizada pelos *bots* com melhor resultado é o uso das chamadas *rotas de abastecimento*, na qual o *bot* mantém um fluxo de naves saindo dos planetas mais afastados do inimigo para os planetas mais próximos, de modo que ele tem mais naves na região de disputa. Outra melhoria seria o ataque e defesa em conjunto, onde um planeta não tem naves suficientes para conquistar ou defender um planeta, mas junto com outros poderia ter. Por fim, o *bot* não avalia as jogadas do adversário, fator que pode ser importante para as decisões das jogadas.

Com mais tempo poderiam ser testados outros métodos de evolução, com outras formas de crossover, outras taxas de mutação ou outras formas de realizar os torneios. Esse fatores deveriam ser mudados para deixar a tempo de convergência maior, já que uma convergência em apenas 44 gerações indica uma certa probabilidade desse resultado ser apenas um máximo local de desempenho do jogador.

6 Conclusão

Um ponto que chamou a atenção é a grande possibilidade de paralelizar a execução do algoritmo genético. Foi possível realizar uma grande quantidade de jogos em paralelo, em tese é possível realizar todos os jogos de uma geração em paralelo.

Outro ponto interessante é que mesmo a população inicial sendo gerada aleatoriamente, de modo que os pesos não tinham sentido lógico, ao decorrer do tempo foram sendo criadas novas populações que acabaram por superar o *bot* cujos pesos foram selecionados manualmente.

A execução da evolução genética através de uma seleção baseada em torneio entre os indivíduos da população, considerando que todos os indivíduos trabalham sob um mesmo arcabouço, pode levar a uma população limite com boas características mas suscetível àquelas estratégias às quais não foi exposta, em analogia ao que realmente ocorre na natureza. É difícil dizer se o indivíduo gerado ao final do processo é robusto.