Rodrigo Mendonça da Paixão
Lucas Teles Agostinho

Titulo a definir

São Paulo – Brasil

2016

Rodrigo Mendonça da Paixão
Lucas Teles Agostinho

Titulo a definir

Pré-monografia apresentada na disciplina Trabalho de Conclusão de Curso I, como parte dos requisitos para obtenção do título de Bacharel em Ciência da Computação.

Centro Universitário Senac

Bacharelado em Ciência da Computação

Orientador: Eduardo Heredia

São Paulo – Brasil

2016

Lista de abreviaturas e siglas

GA	Algoritmos Genéticos

Sumário

1 INTRODUÇÃO 4
1.1 Motivação 4
1.2 Escopo 4
1.3 Justificativa 4
1.4 Objetivos 4
1.5 Método de trabalho 4
1.6 Organização do trabalho 4

2 REVISÃO DE LITERATURA 5
2.1 Busca de caminhos 5
2.1.1 O Algoritmo A* 5
2.1.2 Aplicações 7
2.2 Algorítimos genéticos 8
2.2.1 Aplicações 8
2.3 Algoritmos de busca paralelo 9
2.4 Algorítimos genéticos para busca de caminhos 9
2.5 Algorítimos genéticos paralelos ou distribuídos para busca de caminhos 10

3 PROPOSTA 13

4 CRONOGRAMA 14

REFERÊNCIAS 15

1 Introdução

1.1 Motivação

O problema de buscar o melhor caminho entre dois pontos tem uma grande importância em problemas da engenharia e ciência, tais como rotear o trafico de telefone, navegar por um labirinto ou mesmo definir o layout de trilhas impressas em uma placa eletrônica.

Busca de caminhos também tem uma grande importância no âmbito dos jogos digitais, aonde um jogador compete ou coopera com uma inteligencia artificial e é preciso chegar ao seu destino de forma competente, como por exemplo, jogos de tiro em primeira pessoa ou de estrategia em tempo real.

O valor de entretenimento do jogo pode ser drasticamente reduzido, quando os personagens não podem atravessar um mapa complexo de forma competente, podendo afetar a experiencia de jogo ao deixar visível para o jogador a sua incapacidade de lidar com a busca de caminho de forma satisfatória.

Ainda é comum em jogos digitais termos mais de um agente de busca de caminho ao mesmo tempo no mesmo cenário, podendo ser muitas vezes muito custoso computacionalmente falando. Por isso vários desenvolvedores de jogos tem juntado esforços para desenvolver soluções de busca de caminho em ambientes de recursos escassos. (PONTEVIA, 2008)

1.2 Escopo

1.3 Justificativa

1.4 Objetivos

1.5 Método de trabalho

1.6 Organização do trabalho

2 Revisão de Literatura

2.1 Busca de caminhos

2.1.1 O Algoritmo A*

O algorítimo A* é um dos mais populares soluções no ramo de busca de caminhos, o algoritmo garante achar o menor caminho entre dois pontos (HART; RAPHAEL, 1968), porem gera uma grande arvore de busca nos processos, consumindo muito tempo e memoria, é comum haver modificações no algorítimo para explorar uma arvore de busca menor, diminuindo o tempo para achar um caminho sacrificando a garantia de se encontrar o melhor caminho no final. (BOTEA, 2004).

O algoritmo A* em sua forma tradicional, utiliza a formula heurística f(n) = g(n) + h(n), onde g(n) é o custo para chegar ao nó n, e h(n) é o custo estimado para atingir o nó de destino a partir do nó n. Este sempre deve ser menor ou igual a distancia real entre o ponto n e o destino. Para cada iteração sobre os vizinhos do nó atual é calculado o f(n) e adicionado em uma lista de nós abertos (A), mantendo uma referencia do no que serviu de origem para chegar nele, caso o nó ja esteja na lista e o valor de f(n) novo for menor, o valor de f(n) e o nó de origem é substituído pelo novo. Depois é verificado na lista o menor valor de f(n), este é removido, adicionado a lista de nos fechados (F) e a partir desse ponto, ele se torna o nó atual, quando o nó atual é o mesmo nó de destino o algorítimo retorna o caminho encontrado, assim podemos encontrar a solução ideal. (HART; RAPHAEL, 1968)

Podemos aplicar o algoritmo A* em um Grafo direcionado ponderado por exemplo. (Figura 1)

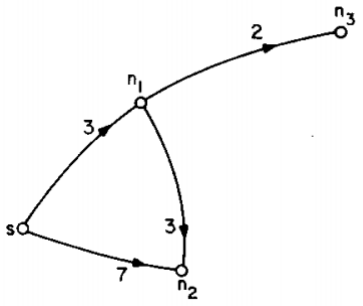

Figura 1 – Grafo para busca (HART; RAPHAEL, 1968)

Figura 1 Consiste em um nó inicial s e mais três outros nós (n1,n2,n3). As arestas contem a direção e o custo do trajeto. Se partirmos do algorítimo A* para produzir um subgrafo do melhor caminho, partindo de s podemos ir para n1 e n2, os valores de g(n1) e g(n2) são respectivamente 3 e 7. Supondo que A* expanda n1, sucedido por

n2 e n3, nesse ponto g(n3)=3+2=5,o valor de g(n2) é diminuído pois um caminho de menor custo foi encontrado 3+3=6, o valor de g(n1) continua sendo 3. O próximo passo seria estimar o h(n), podem essa função vai depender muito do domínio do problema, muitos problemas de encontrar o menor caminho entre dois pontos de um grafo possuem alguma "informação física"que pode ser usada de alguma forma para estimar o h(n), podemos em nosso exemplo, se definirmos como cidades ligadas por estradas, h(n) poderia ser a distancia aérea da cidade n até a cidade objetivo, essa distancia seria menor do que qualquer estrada da cidade n até o objetivo (HART; RAPHAEL, 1968). Em jogos digitais e buscas em tempo real é muito comum trabalhar em cima de mapas em forma de matrizes bidimensionais, esse sera o foco dos algoritmos que trataremos, nesses casos fica simples tratarmos o h(n) com a distancia euclideana ou manhatam entre n e o destino. (BJöRNSSON et al., 2005)

Quando definimos o mapa de busca em uma matriz bidimensional NxM, situação comum em jogos digitais e problemas de busca em tempo real (GRAHAM, 2003) (SANTOS, 2012) precisamos definir os custos de movimentação entre os vizinhos do ponto n, e a função heurisca h(n). O custo de movimentação na grade pode variar basicamente em dois tipos, levando as diagonais em consideração ou não.

O modelo em *tile*, onde o movimento do agente está restrito aa quatro direções ortogonais com custo 1, e *octile*, onde o agente pode ainda mover-se na diagonal com custo equivalente ao valor da hipotenusa de um triangulo retângulo com catetos igual a 1, sendo assim $\sqrt{2} \approx 1.4$. É comum para fim de simplificar os cálculos multiplicar os valores de custo por 10 ou 100 como pode ser visto na figura 2.

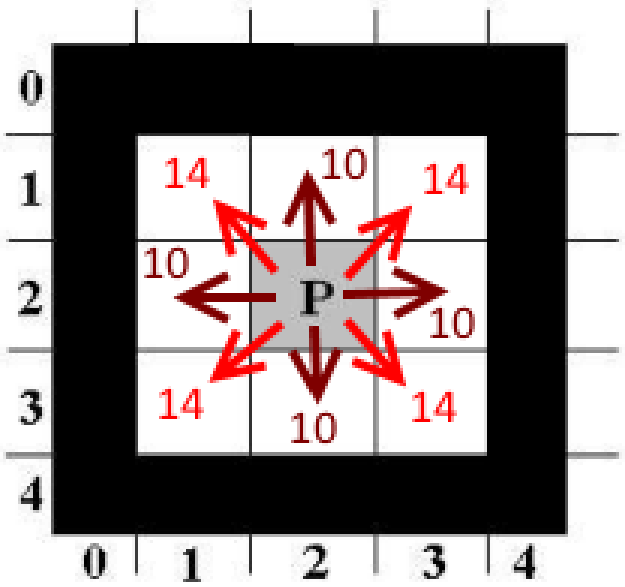

Figura 2 – Custo de movimentação

Uma limitação do A*, é o fato que ele requer uma grande quantidade de recursos da

CPU, caso haja muitos nós a pesquisar como é o caso em grandes mapas que sãp populares em jogos recentes, isso pode causar pequenos atrasos. Esse atraso pode ser agravado caso haja múltiplos agentes IA ou quando o agente tem que se mover de um lado do mapa para o outro

Este alto custo de recursos da CPU pode fazer com que o jogo congele até que o caminho ideal seja encontrado. Game designers tentam superar esses problemas realizando ajustes nos jogos, de forma a tentar evitar estas situações (CAIN, 2002).

Podemos observar o motivo do alto custo, a heurística fornece uma expansão considerável dos nós, que são mantidos na memoria durante todo o processamento. (Figura 3)

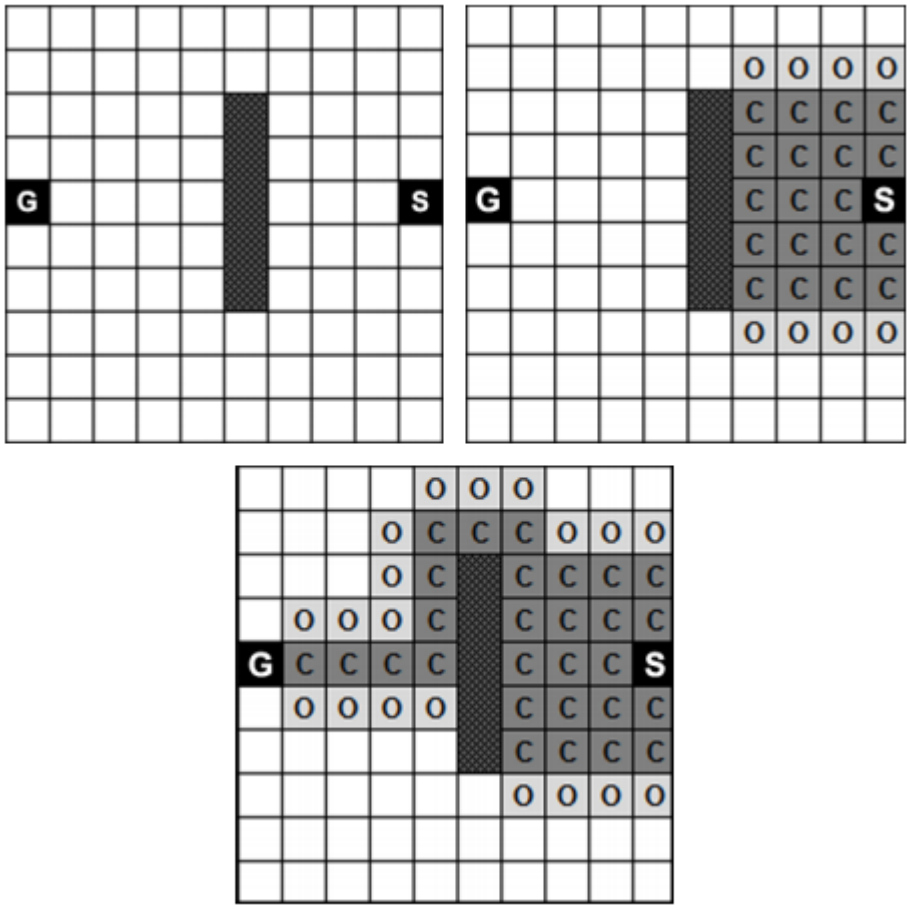

Figura 3 – Exemplo de A (SANTOS, 2012)*

2.1.2 Aplicações

Aplicações para os algoritmos de busca podem ser diversos, como por exemplo, em um simulador de carro corrida (WANG; LIN, 2012). Utilizando o algoritmo do A* com duas modificações para encontrar o melhor caminho enquanto evita os obstáculos entre o ponto de inicio e o ponto de destino. A primeira modificação consiste em utilizar o Teorema de Triângulos de Pytagoras, onde primeira calcula a distancia entre dois pontos(1

e 2), verifica se existe algum obstaculo dentre eles, se existir, utiliza o terceiro ponto para calcular a Hipotenusa e verifica se existe obstaculo entre a hipotenusa, se não existir, remove o ponto 2 e começa a considerar o caminho do ponto 1 para o ponto 3. A segunda modificação consiste em somente ir para frente, isso significa, procura somente os pontos a frente do carro, direita, esquerda e frente, simulando um controle de carro.

O projeto utiliza 3 pistas reais de corrida da Formula 1, Peru, Itália e Hungria, sendo cada pista, uma imagem de escala 1280x782 pixels retiradas do site oficial da Formula 1. O Carro é implementado em Microsoft XNA Game Studio, plataforma usada para desenvolver jogos para Windows Phone, Xbox e Windows. As Imagens das pistas são modificadas para serem mapas de detecção de colisão, a pista é pintada de preto, indicando onde o carro pode andar e o resto pintado de branco indicando os o carro não pode andar. O carro tem o tamanho de 18x12 pixels e uma movimentação de 2 pixels por segundo. Como o XNA trabalha com uma taxa de quadros de 60 quadros por segundo, o carro se movimenta a 120 pixels por segundo.

Os resultados da primeira modificação conseguiu economizar ciclos de CPU, reduzindo o numero de pontos indicadores da pista em 97%. A segunda modificação tem a vantagem de obter o tempo de volta mais curto, por que reduz o numero de nós do algoritmo A* de 4 para 3 nós]. A desvantagem é que o carro pode balançar em curvas acentuadas. Em geral, as modificações garantiram uma melhoria em performance, economizando o numero de ciclos de CPU.

2.2 Algorítimos genéticos

2.2.1 Aplicações

Existem vários aplicações para os algoritmo geneticos, por serem uma inteligencia artificial não supervisionada, de rápido aprendizado e podendo ser paralelizado.

O modelo m-PRC(Problema de Rotas de Cobertura multi-veiculo) é uma aplicação de algoritmos geneticos para construção de rotas em uma região mapeada, para encontrar uma boa distribuição de viaturas para patrulhamento urbano usado por departamentos de segurando como a policia, guardas municipais ou segurança privada. O Modelo é definido como um grafo G=(V U W, E) não direcionado, onde V U W compõem o conjunto de vértices e E o conjunto de arestas, ou seja, o subgrafo induzido por E e um grafo completo cujo conjunto de nós é V. V são todos os vértices que podem ser visitados e é composto pelo subconjunto T, que são os vértices que devem ser visitados por algum veiculo. W é um conjunto de vértices onde todos os M veículos devem passar. M é o numero de rotas de veículos que começam no vértice base V0. O m-PRC atribui o conjunto de m rotas de veículos com as restrições: Todas as m rotas de veículos começam e terminam na base

V0, Tem exatamente m rotas, Cada vértice de V pertence a no máximo uma rota, Cada vértice de T pertence a exatamente uma rota, com exceção a base, Cada vértice de W deve ter uma rota que passa por ele e em uma distancia C de um vértice V visitado, O modulo da diferença entre o número de vértices de diferentes rotas não pode exceder um determinado valor R.

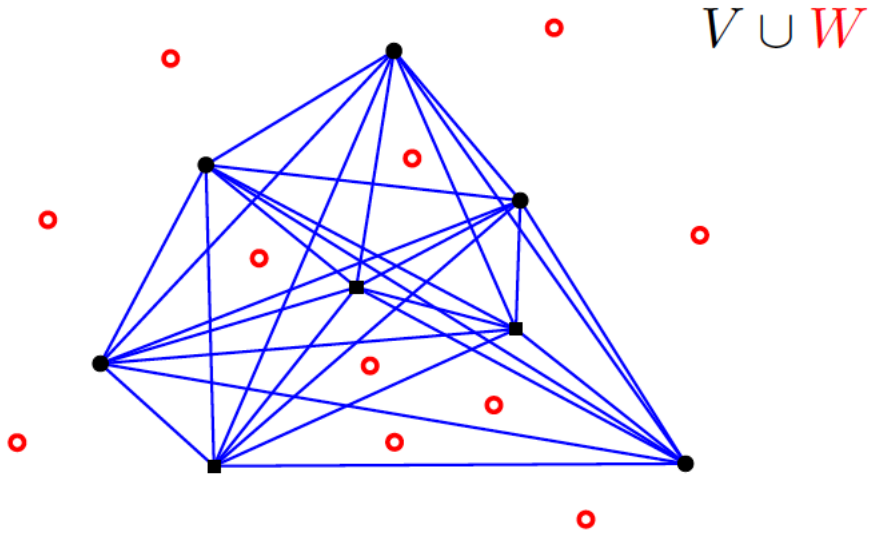

Figura 4 – Exemplo de gráfo não direcionado para V U W. (OLIVEIRA, 2009)

2.3 Algoritmos de busca paralelo

2.4 Algorítimos genéticos para busca de caminhos

Buscando melhorar a eficiência dos algoritmos de busca, foram criadas formas hibridas, levando os algoritmos genéticos analisarem todo o percusso,ajudando o algoritmo de busca em momentos que a próxima ação é incerta.

Utilizando o algoritmo de busca BFS (*Best First Search*), foi desenvolvido o modelo PPGA (*Patterned based Pathfinding with Genetic Algorithm*). O modelo utiliza algoritmos genéticos como um modulo para calcular os sub caminhos ao longo do processo de busca, cada vez que o modulo é chamado, herda informações da chamada anterior, tendo um ganha considerado de desempenho. O modulo é chamado, quando nenhum dos valores dados pelo BFS são melhores que o valor atual. Os resultados da analise do PPGA foram bons, por encontrar boas soluções em um curto período de tempo, mostrando ser muito bom para mapas onde o percurso segue um padrão, mas perdendo para o BFS para mapas mistos ou que não seguem o mesmo padrão ao longo o percurso, considerando o modelo como promissor, indicando que mudanças nos parâmetros do algoritmo genéticos, pode melhorar o desempenho dos testes (SANTOS, 2012).

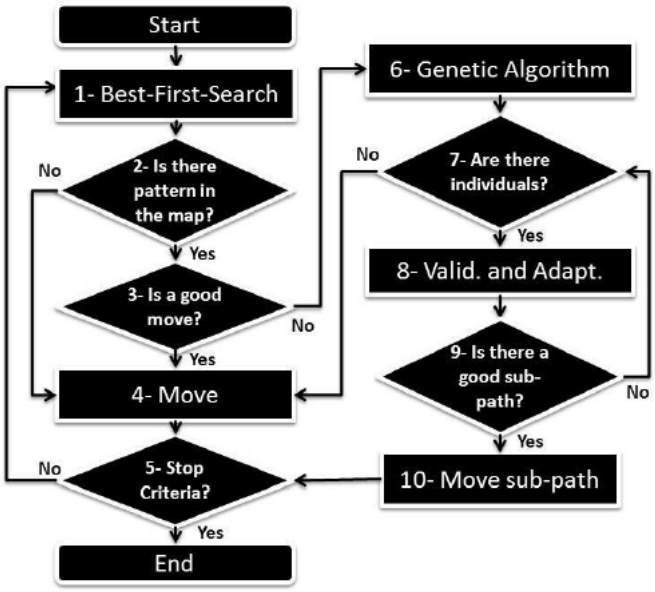

Figura 5 – Fluxograma do Modelo (SANTOS, 2012)

O modelo GAMMA(*Genetic Algoritmo Manufactured Maneuvering Algorithm*), consiste em uma modificação do A* de forma que o caminho total é separado em caminhos mais curto. O algoritmo prepara o primeiro caminho e calcula o caminho mais curto entre eles, a posição final do resultado é usada como inicial para o próximo calculo, com isso, permitindo que o modelo GAMMA procure caminhos em tempo real por que se o caminho for alterado a sua leitura é parcial do caminho completo.(LEIGH; MILES, 2007)

2.5 Algorítimos genéticos paralelos ou distribuídos para busca de caminhos

Foi demonstrado que algoritmos genéticos paralelos são eficientes para a resolução de problemas de busca de caminho, tal como o clássico problema do caixeiro viajante, que consiste em dado um numero finito de cidades com seus custos de viagem entre elas, deve-se encontrar o caminho mais curto para viajar entre todas as cidades e voltar ao ponto inicial. O problema pode ser representado pelo modelo de um grafo direcionado ponderado, aplicando a mesma ideia, o problema seria encontrar o caminho de menor custo para percorrer todos os nós, de maneira análoga, as cidades seriam os nós e a distancia entre elas, o peso das arestas (D.LOHN SILVANO P. COLOMBANO, 2000)(ALAOUI; EL-GHAZAWI, 2000)(MUHLENBEIN, 2000).

Os algoritmos genéticos exigem apenas o valor dado por uma função objetivo como parâmetro e mesmo sobre espaços de busca grandes, tem uma convergência rápida. Por causa do processo associado, agrega uma visão mais global do espaço de busca na prática de otimização e possuem uma fácil paralelizar por causa da independência dos

seus processos. Em comparação com as técnicas de busca mais comuns,que requerem informações derivadas, continuidade do espaço de busca ou conhecimento completo da função objetiva.(MOLE, 2002).

A solução para este tipo de problema pode requer uma quantidade grande de processamento. Uma boa solução seria dividir o processamento do problema em pequenas partes e distribuir cada parte para um processador a parte, trabalhando de forma distribuída ou paralela. Vários modelos para essa finalidade foram propostos.

Um modelo interessante para paralelização seria o de mestre-escravo, onde o mestre fica responsável na manutenção da população e execução dos operadores genéticos. A avaliação dos melhores indivíduos é distribuída para os demais escravos, O mestre envia um indivíduo a cada um dos escravos subjacentes. Cada escravo realiza a interpretação do problema, aplica a função de cálculo para a escolha dos melhores indivíduos e envia seus resultados ao mestre, que executa seleção dos indivíduos e a geração da nova população, repetindo o processo como um todo. Essa estrutura teve implicação satisfatória para a automação de design de circuitos eletrônicos. (D.LOHN SILVANO P. COLOMBANO, 2000)

Outra forma de trabalhar com o modelo de mestre escravo, seria definir que cada um dos nós escravos subjacentes fica responsável por sua própria população. O nó central mestre, cria as populações iniciais e as distribui para os nós escravos. Cada nó escravo processa a evolução da população por um determinado número de gerações e então a submete ao mestre. O mestre então seleciona os melhores indivíduos dentre todas as populações dos nós escravos e os distribui novamente. Em cada nó escravo, os novos indivíduos distribuídos pelo mestre são inseridos na população corrente e o processo de evolução recomeça. A migração entre os escravos, que é controlado pelo nó mestre, implementa o mecanismo que regula a velocidade da convergência e oferece os meios de escape dos mínimos locais. Entretanto a migração das populações dos nós escravos para o mestre e vice versa pode impor um certo grau de sobre carga, dependente do meio de comunicação entre os nós. Esse modelo obteve sucesso no mapeamento de tarefas em maquinas paralelas. (ALAOUI; EL-GHAZAWI, 2000)

Podemos partir do ponto que cada indivíduo é o responsável por encontrar e reproduzir com um parceiro em sua vizinhança. O controle de seleção e reprodução se espalha pela população e o algoritmo deixa de ser centralizado em um mestre, com isso, diminui o grau de sincronização e facilita a paralelização. O processo do algoritmo é definir uma representação genética para o problema e criar a estrutura de vizinhança e sua população inicial. Cada indivíduo faz uma busca em sua vizinhança e seleciona um parceiro para a reprodução. Uma geração descendente é criada com o operador genético resultante. (MUHLENBEIN, 2000)

Podemos observar alguns problemas nos modelos apresentados (MOLE, 2002), no

modelo de (D.LOHN SILVANO P. COLOMBANO, 2000) existe problema em explorar o paralelismo no calculo de verificação dos indivíduos não explorando para a reprodução e mutação. No modelo de (MUHLENBEIN, 2000), tem a possibilidade de utilizar vários métodos de busca de indivíduos da mesma população, sendo úteis em casos que a eficiência dos métodos de busca se mostram dependentes da instancia do problema. O modelo (ALAOUI; EL-GHAZAWI, 2000), por todos os escravos devem enviar para o nó mestre, demanda uma grande capacidade de processamento no nó mestre, e proporciona a divisão das populações em pequenas ou de médio porte.

(MOLE, 2002) desenvolveu seu próprio modelo, utilizando o modelo de (ALAOUI; EL-GHAZAWI, 2000) como inspiração. O modelo segue o conceito mestre-escravo, o mestre crias as populações e distribui a cada uma delas, os conjuntos de genes e parâmetros iniciais. O mestre é utilizado para a troca de indivíduos entre as populações, mantendo um indivíduo de cada população até serem substituídos por um melhor e envia esses indivíduos para as populações que não seja a sua de origem. As populações são independentes, gerando seus indivíduos iniciais com base nos genes enviados pelo mestre, aplicando seus próprios operadores de evolução e a população que determina os parceiros dos indivíduos.

3 Proposta

4 Cronograma

Para modelar a proposta descrita, sera seguido um cronograma semanal, este ira descrever semana a semana as tarefas que devem ser realizadas de forma a se concluir o trabalho.

Semana	**Atividade**
1ª	A Definir
2ª	A Definir
3ª	A Definir
4ª	A Definir
5ª	A Definir
6ª	A Definir
7ª	A Definir
8ª	A Definir
9ª	A Definir
10ª	A Definir
11ª	A Definir
13ª	A Definir
14ª	A Definir
15ª	A Definir
16ª	A Definir
17ª	A Definir
18ª	A Definir

Referências

ALAOUI, O. F. S. M.; EL-GHAZAWI, T. A parallel genetic algorithm for task mapping on parallel machines. 2000. Citado 3 vezes nas páginas 10, 11 e 12.

BJöRNSSON, Y. et al. Fringe search: beating a* at pathfinding on game maps. In: *In Proceedings of IEEE Symposium on Computational Intelligence and Games*. [S.l.: s.n.], 2005. p. 125–132. Citado na página 6.

BOTEA, M. M. A. Near optimal hierarchical path-finding. *Journal of Game Development, Vol. 1*, p. 7–28, 2004. Citado na página 5.

CAIN, T. *Practical Optimizations for A**. [S.l.: s.n.], 2002. Citado na página 7.

D.LOHN SILVANO P. COLOMBANO, G. L. H. t. D. S. J. Parallel genetic algorithm for automated electronic circuit design. 2000. Citado 3 vezes nas páginas 10, 11 e 12.

GRAHAM, H. M. e. S. S. R. Pathfinding in computer games. *The ITB Journal*, v. 4, 2003. Citado na página 6.

HART, N. J. N. P. E.; RAPHAEL, B. A formal basis for the heuristic determination of minimum cost paths. *IEEE Transactions on Systems, Science, and Cybernetics*, SSC-4, p. 100–107, 1968. Citado 2 vezes nas páginas 5 e 6.

LEIGH, S. J. L. R.; MILES, C. Using a genetic algorithm to explore a*-like pathfinding algorithms. 2007. Citado na página 10.

MOLE, V. L. D. Algoritmos genéticos – uma abordagem paralela baseada em populações cooperantes. 2002. Citado 2 vezes nas páginas 11 e 12.

MUHLENBEIN, H. Evolution in time and space - the parallel genetic algorithm. 2000. Citado 3 vezes nas páginas 10, 11 e 12.

OLIVEIRA, W. A. de. Algoritmo genético para o problema de rotas de cobertura multiveículo. 2009. Citado na página 9.

PONTEVIA, P. Pathfinding is not a star. *Autodesk, white paper*, p. 1–6, 2008. Citado na página 4.

SANTOS, A. F. V. M. e. E. W. G. C. U. O. Pathfinding based on pattern detection using genetic algorithms. *SBC - Proceedings of SBGames*, 2012. Citado 4 vezes nas páginas 6, 7, 9 e 10.

WANG, J.-Y.; LIN, Y.-B. Game ai: Simulating car racing game by applying pathfinding algorithms. *International Journal of Machine Learning and Computing*, v. 2, 2012. Citado na página 7.