Lucas Teles Agostinho
Rodrigo Mendonça da Paixão

Uma abordagem sobre a utilização de paralelismo em algoritimos geneticos aplicados em busca heuristica de caminho

São Paulo – Brasil

2016

Lucas Teles Agostinho
Rodrigo Mendonça da Paixão

Uma abordagem sobre a utilização de paralelismo em algoritimos geneticos aplicados em busca heuristica de caminho

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

AG	Algoritmos Genéticos
A*	AStar ou A Estrela
IA	Inteligência Artificial
CPU	Central Processing Unit]
PCV	Problema do Caixeiro Viajante
BFS	Best-first search

Sumário

1 Introdução

1.1 Motivação

O problema de buscar o melhor caminho entre dois pontos tem uma grande importância em areas como 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, onde 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 têm juntado esforços para desenvolver soluções de busca de caminho em ambientes de recursos escassos.

(PONTEVIA, 2008)

Muitas abordagens para melhorar o desempenho ou diminuir o custo de métodos de busca de caminho tem sido desenvolvidos (SANTOS, 2012) (POLLACK, 1960) (CAIN, 2002) (MILLER, 2010).

Ainda temos problemas de alto custo computacional, que para serem minimizados em alguns casos, acabamos sacrificando a certeza de melhor caminho por um melhor desempenho(BOTEA, 2004)(KORF, 1993). Por isso iremos fazer uma análise e propor um modelo para otimizar alguma limitação do modelo, visando em principal o âmbito dos jogos digitais.

Qualquer problema que possa ser formulado em busca em grafos pode solucionado a partir de busca heuristicas. O exemplo mais comum problema que podem ser resuzido é o PCV, outro exemplo tipico é quebra cabeça de delizar, que consiste em quadro que contem quadrados numerados e uma posição vazia (ver 2). As operações permitidas são deslizar qualquer peça horizontalmente ou verticalmente adjacente a peça vazia para a posição vazia, o objetivo é rearranjar as peças a partir de uma confiruação randomica com

menor quantidade de movimentos. Encontrar a solução otima para esse problema ou para o PCF é NP-Completo. (KARP, 1972) (**??**)

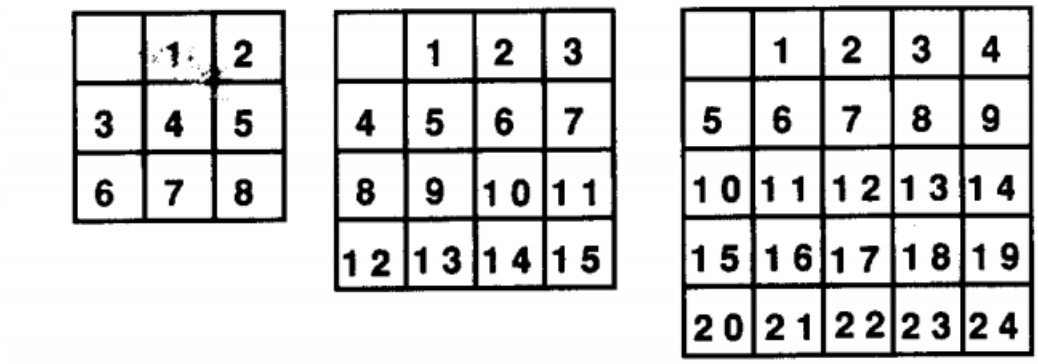

Figura 1 – Quebra cabeça de deslizar (KORF, 1993)

Este cenário é a principal motivação deste trabalho que consiste em propor, implementar e mensurar resultados de uma solução para busca de melhor caminho entre dois pontos.

1.2 Objetivos

Este trabalho tem como objetivo verificar a aplicabilidade de paralelismo de AG para melhoria de precisão em busca heurística de caminhos. O ponto central é a definição de uma estrutura capaz de explorar as vantagens demonstradas no uso de AG com computação paralela e aplica-las em um modelo de busca heurística de caminho que use AG para redução de custo de memória e processamento, ja que para isto, estes tendem a sacrificar a certeza de encontrar a menor rota.

Implementar o modelo de busca heurística com AG e mensurar os resultados.

Aplicar um modelo de paralelização de AG.

Combinar o modelo paralelo com o de busca e mensurar os resultados.

1.3 Método de trabalho

Primeiramente sera desenvolvido uma bateria de testes de caminhos, com intuito de terem diferentes tamanhos, levando em cosideração mapas com padrões de repetição e sem padrões de repetição, os mesmos serão modelados de forma bidimensional em arquivos de texto com caracteres para definir o ponto inicial (S), final (E), obstáculos (#) e caminho livre (.).

Será implementada a busca heuristica A* utilizando a linguagem C# .NET, levando

em consideração os arquivos de teste desenvolvidos como entrada, retornando se existe um caminho entre o ponto inicial e final, quanto tempo levou para encontrar o caminho e consumo de memoria para chegar na solução.

Também será implementado um AG para utilizar em conjunto ao A*, também utilizando a linguagem C# .NET, implementando o mesmo método proposto por Ulysses O. Santos (SANTOS, 2012). A implementação receberá arquivos de testes de entrada contendo os mapas, os mesmos utilizado para testar a implementação do A* anteriormente, tambem retornando o tempo e consumo de memoria para chegar na solução. Os parâmetros utilizados para o AG serão uma população inicial de 4 indivíduos, com o critério de geração de descendentes em 50%, com o critério de mutação em 25% e será calculada a aptidão para 12 indivíduos.

Por ultimo, será implementado uma modificação do AG para paraleliza-lo, como proposto por (ALAOUI; EL-GHAZAWI, 2000). Serão utilizadas N *threads* para a separação das populações, ou seja, trabalhando com varias populações ao mesmo tempo, onde cada *thread* fica responsavel por uma, para depois mesclar os melhores individuos entre elas, esse sistema ira receber os mesmo arquivos de testes utilizados nas implementações anteriores, retornando informações de tempo e consumo de memória.

Nas três implementações todos os testes serão executados 100 vezes e será calcula a média do tempo e consumo de memoria de cada um. Todos os dados coletados serão comparados em forma de tabelas e gráficos, para demostrar os ganhos e perdas relativos de cada abordagem.

1.4 Organização do trabalho

2 Revisão de Literatura

Nesse capitulo é feita uma revisão no estado da arte de diferentes formas de utilizar os algoritmos de busca, algoritmos geneticos, paralelismo ou uma união de todas ao mesmo tempo.

2.1 Busca Heuristica

Busca heurística é um mecanismo geral de solução de problemas no ramo de IA. A sequência de passos necessários para solução da maioria dos problemas de IA usualmente não é conhecida, mas deve ser determinada por uma exploração de alternativas atraves de mecanismos de tentativa e erro. Um problema busca típico é o PCV, que consiste em encontrar o caminho mais curto que as visite cada cidade de um conjunto cidades e retorne para a cidade de partida.

2.1.1 *Best-first search*

É um algoritmo de busca heurística geral. Ele mantém uma lista aberta contendo os nós ainda não visitados de uma árvore que gerada, e uma lista fechada dos nos que já foram expandidos e visitados. Cada nó tem um valor de custo, em cada ciclo um nó de menor custo da lista aberta é expandido, cada filho do nó é avaliado por uma função de custo e adicionado na lista aberta, e o nó pai é adicionado a lista de nos fechados.

A lista aberta contem apenas o nó inicial e o algoritimo termina quando encontrar o nó de destino para expansão.(KORF, 1993)

Existem muitas implementações do BFS, tais como *Dijkstra's* ou A*, o que os diferencia basicamente é como é implementada sua função de custo.

2.1.2 O Algoritmo A*

O algorítimo A* é um dos mais populares soluções no ramo de busca de caminhos, um desses motivos é que sempre garantindo achar o menor caminho entre dois pontos (HART; RAPHAEL, 1968), porém, gera uma grande árvore de busca nos processos, consumindo muito tempo e memória. É comum haver modificações no algoritmo para explorar uma árvore 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 fórmula 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 distância real entre o ponto n e o destino, ou seja nunca pode ser super-estimado. 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 referência do nó 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 da Figura 2.

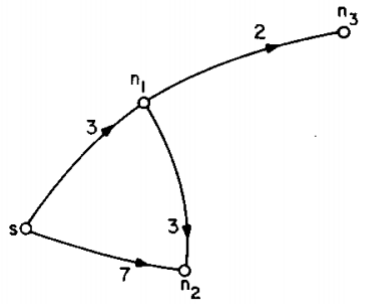

Figura 2 – Grafo para busca (HART; RAPHAEL, 1968)

O grafo 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 algoritmo 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 será o foco dos algoritmos que trataremos, nesses casos fica simples tratarmos o h(n) com a distância Euclideana ou Manhattan 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) (MACHADO et al., 2011), precisamos definir os custos de movimentação entre os vizinhos do ponto n, e a função heurística 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 a 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 triângulo 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 3.

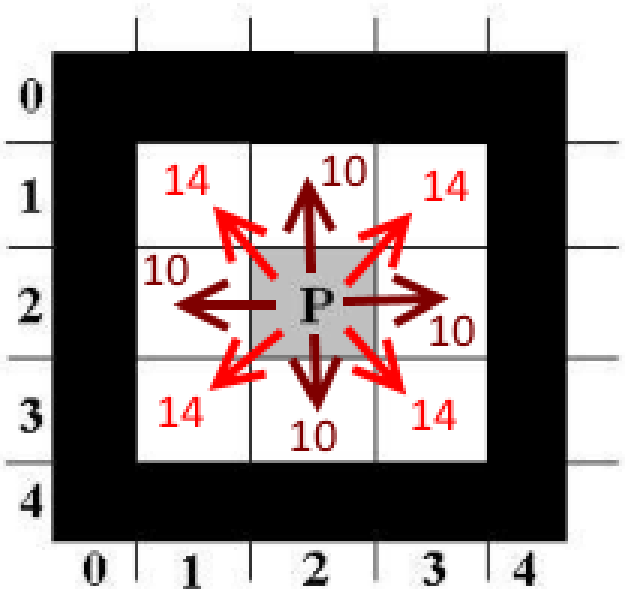

Figura 3 – Custo de movimentação

Uma heurística famosa e muito utilizada é conhecida como distancia manhattan, usada principalmente em ambientes de grades, ela estima a distancia real entre os nós pela soma do deslocamento vertical e horizontal entre dois pontos, ignorando qualquer obstaculo. A distância manhattan garante a solução com mais baixo custo, pois cada nó deve caminhar pelo menos, a sua distância manhattan, e o movimento é dado de um nó por vez (KORF, 2000).

Ainda pode ser facilmente modificado para levar em conta o produto vetorial entre o vetor do ponto inicial e o vetor de objetivo, resultando em um caminho "reto"entre dois nós. (PATEL, 2010).

Ainda pode ser realizada uma alteração no algoritmo de distancia manhattan para levar em conta as diagonais, este fica chamado de "distancia diagonal", adicionando um custo de movimentação diagonal diferente ao de movimentação adjacente.

Também pode ser utilizada como função heurística a distancia euclideana, no entanto, neste caso teremos problemas com o uso de A*, pois a função de custo g não

irá corresponder a função heurística h. Ja que a distância euclidiana é mais curta do que Manhattan ou a distância diagonal, você ainda terá os caminhos mais curtos, mas o A* levará mais tempo para ser executado (PATEL, 2010).

Podemos verificar na Figura 4, exemplos de distancia manhattan comparado com a distancia euclideana.

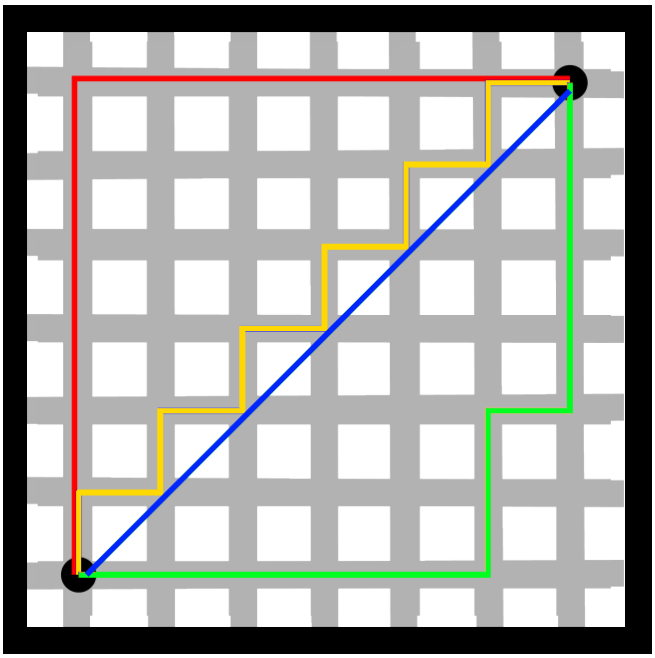

Figura 4 – Ilustração de diferença entre distancia Manhatam e euclideana

Figura 4 As linhas de cor vermelho, verde e amarelo tem o mesmo tamanho (12), e representam a distancia manhattan, a amarela leva em consideração o produto vetorial. A linha azul representa a distancia euclideana $\sqrt{72} \approx 8.485$

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 são populares em jogos recentes, isso pode causar pequenos atrasos. Esse atraso pode ser agravado caso haja múltiplos agentes de inteligência artificial 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).

Pelo motivo do alto custo, a heurística fornece uma expansão considerável dos nós, que são mantidos na memória durante todo o processamento. A Figura 5 mostra a expansão da heurística.

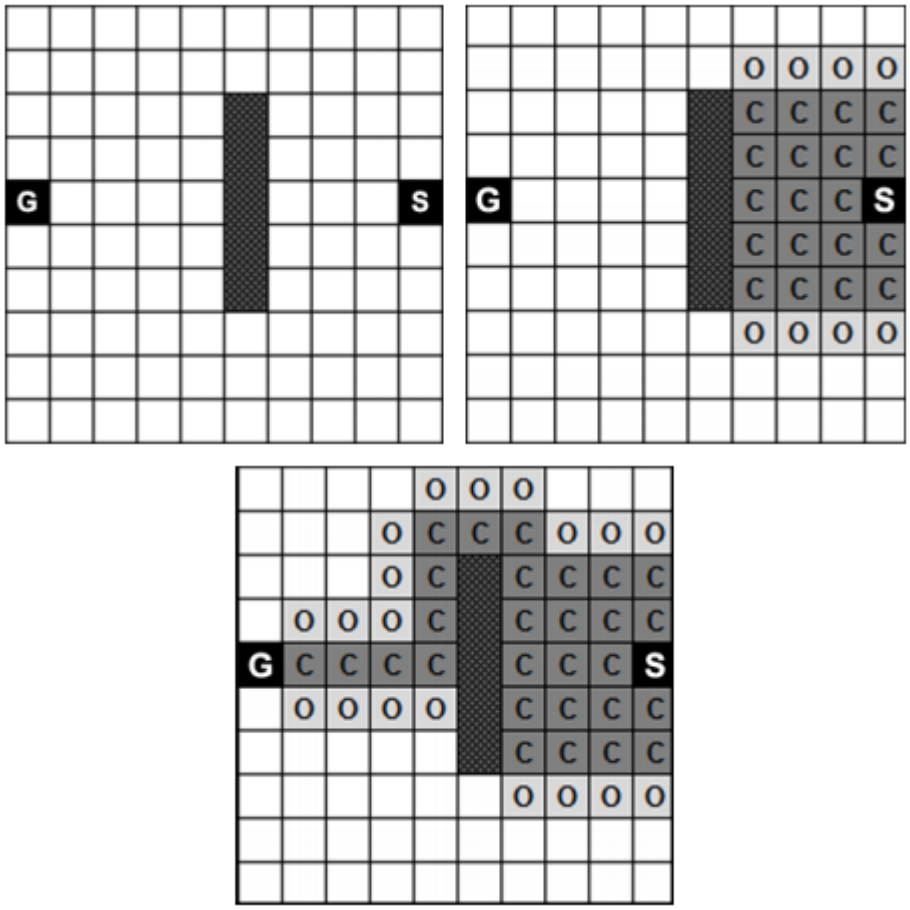

Figura 5 – Exemplo de A (SANTOS, 2012)*

É comum no meio dos jogos digitais termos mais de um agente rodando ao mesmo tempo, um otimização para esse problema, proposta por Ko- Hsin, Cindy Wang e Adi Botea, examina o planejamento de caminhos multi-agente. Eles propõem um algoritmo chamado de FAR (*Flow Annotation Replanning*) para combater os problemas técnicos envolvidos em pesquisas globais por várias entidades. FAR implementa restrições de fluxo para evitar colisões frontais, para assim reduzir o espaço de busca que uma entidade deve considerar quando planejar a sua rota. Esse método tem a possibilidade de entrar em *deadlock*, um estado de bloqueio, porem o algoritmo é capaz de contornar esse problema (WANG; BOTEA, 2008).

A arte de busca de caminhos está longe de um trabalho perfeito e muito ainda precisa ser feito (PONTEVIA, 2008).

2.1.3 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 Pitagoras, onde primeira calcula a distância entre dois pontos(1 e 2), verifica se existe algum obstáculo dentre eles, se existir, utiliza o terceiro ponto para calcular a Hipotenusa e verifica se existe obstáculo 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 de busca paralelo

O problema de busca de caminho pode ser facilmente quebrado em duas maneiras (MILLER, 2010).

A primeira abordagem seria quebrar o problema em dois, iniciando a busca de ambos os nós, inicial e final simultaneamente, Esta variação do A*, também conhecida como Pesquisa Bidirecional, permite que um problema seja dividido em dois núcleos, mas não escala facilmente com um maior numero de núcleos ou com um problema de maior tamanho. Esta abordagem requer dois núcleos livres por busca, e não seria capaz de executar em um ambiente que não ofereça no minimo essa estrutura. (CHANG, 2009).

A segunda abordagem seria tratar cada busca de caminho como uma unidade individual e distribuir cada uma para um núcleo. Esta abordagem funciona facilmente com qualquer número de núcleos e é possivelmente a forma mais eficaz de distribuição de grandes conjuntos de dados (CHAMPANDARD, 2010) (MILLER, 2010).

2.3 Algoritmos genéticos

AG é uma técnica amplamente utilizada de IA, que utilizam conceitos provenientes do principio de seleção natural para abordar uma ampla serie de problemas, geralmente de adaptação.

2.3.1 Funcionamento

Inspirado na maneira como o seleção natural explica o processo de evolução das especies, Holland (HOLLAND, 1975) decompôs o funcionamento dos AG em sete etapas, essa são *inicialização*, *avaliação*, *seleção*, *cruzamento*, *mutação*, *atualização* e *finalização* ver Figura 6. (LUCAS, 2002)

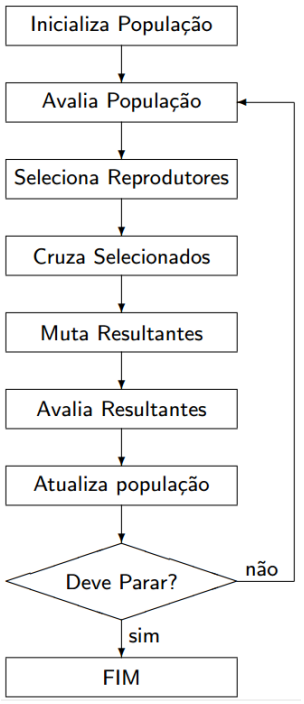

Figura 6 – Estrutura de um AG (LUCAS, 2002)

Inicialização: Criar uma população de possíveis respostas para um problema. É comum fazer uso de funções aleatórias para gerar os indivíduos, sendo este um recurso simples que visa fornecer maior diversidade.

Avaliação: Avalia-se a aptidão das soluções, os indivíduos da população, então é feita uma análise para que se estabeleça quão bem elas respondem ao problema proposto.

A função de avaliação também pode ser chamada de função objetivo. Ela pode variar de acordo com problema, Calcular com exatidão completa o grau de adaptação dos indivíduos pode ser uma tarefa complexa em muitos casos, e se levarmos em conta que esta operação é repetida varias vezes ao longo do processo de evolução, seu custo pode ser consideravelmente alto. Em tais situações é comum o uso de funções não determinísticas, que não avaliam a totalidade das características do individuo, operando apenas sobre uma

amostragem destas.

Seleção: Ela é a responsável pela perpetuação de boas características na espécie. Neste estagio que os indivíduos são escolhidos para posterior cruzamento, fazendo uso do grau de adaptação de cada um é realizado um sorteio, onde os indivíduos com maior grau de adaptação tem maior probabilidade de se reproduzirem. O Grau adaptação é calculado a partir da função de avaliação para cada individuo, determina o quão apto ele esta para reprodução relativo a sua população.

Cruzamento: Características das soluções escolhidas na seleção são recombinadas, gerando novos indivíduos.

Mutação: Características dos indivíduos resultantes do processo de reprodução são alteradas, acrescentando assim variedade a população. A mutação opera sobre os indivíduos resultantes do processo de cruzamento e com uma probabilidade pré-determinada efetua algum tipo de alteração em sua estrutura. A importância desta operação é o fato de que uma vez bem escolhido seu modo de atuar, é garantido que diversas alternativas serão exploradas.

Atualização: Os indivíduos criados no processo de reprodução e mutação são inseridos na população.

Na forma mais tradicional deste a população mantém um tamanho fixo e os indivíduos são criados em mesmo número que seus antecessores e os substituem por completo.

Existem, porém, algumas alternativas, o número de indivíduos gerados pode ser menor ou o tamanho da população pode sofrer variações e o critério de inserção pode variar, por exemplo, nos casos em que os filhos substituem os pais, ou em que estes só são inseridos se possuírem maior aptidão que o cromossomo que sera substituído, ou o manter sempre o conjunto dos n melhores indivíduos.

Finalização: É testado se as condições de encerramento da evolução foram atingidas, retornando para a etapa de avaliação em caso negativo e encerrando a execução em caso positivo.

Os critérios para a parada podem ser vários, desde o número de gerações criadas até o grau de convergência da população atual.

Toda base dos AG se fundamenta nos indivíduos, eles são a unidade básica em qual o algorítimo se baseia, sua função é codificar as possíveis soluções do problema a ser tradado e partir de sua manipulação no processo evolutivo, a partir daí que são encontradas as respostas.

Esses indivíduos precisam de uma representação, essa será o principal responsável pelo desempenho do programa. É comum chamar de *genoma* ou *cromossomo* para se

referir ao individuo. Por essa definição podemos resumir um indivíduo pelos genes que possui, ou seja seu *genótipo*.

Apesar de toda representação por parte do algoritmo ser baseada única e exclusivamente em seu genótipo, toda avaliação é baseada em seu fenótipo, o conjunto de características observáveis no objeto resultante do processo de decodificação dos genes do individuo, ver Tabela 1.

Problema	**Genótipo**	**Fenótipo**
Otimização numérica	0010101001110101	10869
Caixeiro viajante	CGDEHABF	Comece pela cidade C, depois passe pelas cidades G, D, E, H, A, B e termine em F
Regras de aprendizado para agentes	$C_1R_4C_2R_6C_4R_1$	Se condição 1 (C_1) execute regra 4 (R_4), se (C_2) execute (R_6), se (C_4) execute (R_1)

Tabela 1 – Exemplos de genótipos e fenótipos correspondentes em alguns tipos de problemas (LUCAS, 2002)

Para cada indivíduo é calculado o seu grau de adaptação, a partir de uma função objetivo, comumente denotada como:

$$f_O(x)$$

Este vai representar o quão bem a resposta apresentada pelo individuo soluciona o problema proposto.

Também é calculado o grau de adaptação do individuo relativo aos outros membros da população a qual ele pertence, esse é chamado de grau de aptidão, para um individuo x temos seu grau de aptidão denotado pela formula:

$$f_A(x) = \frac{f_O(x)}{\sum_{i=1}^{n} f_O(i)}$$

Sendo n o tamanho da população.

A dinâmica populacional é a responsável pela evolução, ao propagar características desejáveis a gerações subsequentes no processo de cruzamento, enquanto novas são testadas no processo de mutação.

Algumas definições importantes relativo as populações de um AG são:

Geração: É o numero de vezes em que a população passou pelo processo de seleção, reprodução, mutação e atualização.

Média de adaptação:

$$M_A = \frac{\sum_{i=1}^{n} f_O(i)}{n}$$

Grau de convergência: define o qual próxima esta a media de adaptação desta população relativo as anteriores. O objetivo dos AG é fazer a população convergir para uma valor de adaptação ótimo. Um estado negativo que pode ocorrer relativo a esta medida é a *convergência prematura*, a mesma ocorre quando a população converge em uma media de adaptação sub-ótima, e dela não consegue sair por causa de sua baixa diversidade.

Diversidade: Mede o grau de variação entre os genótipos da população. Ela é fundamental para o tamanho da busca. Sua queda esta fortemente ligada ao fenômeno de *Convergência prematura.*

Elite: São os indivíduos mais bem adaptados da população. Uma técnica comum nos AG é p *elitismo*, onde são selecionados k melhores indivíduos que serão mantidos a cada geração.

2.3.2 Aplicações

Existem vários aplicações para os algoritmo genéticos, 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 genéticos 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 não direcionado:

$$G = (V \cup W, E)$$

Onde

$$V \cup W$$

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 V_0.

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 V_0, 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. A Figura 7 mostra o grafo da relação de V com W.

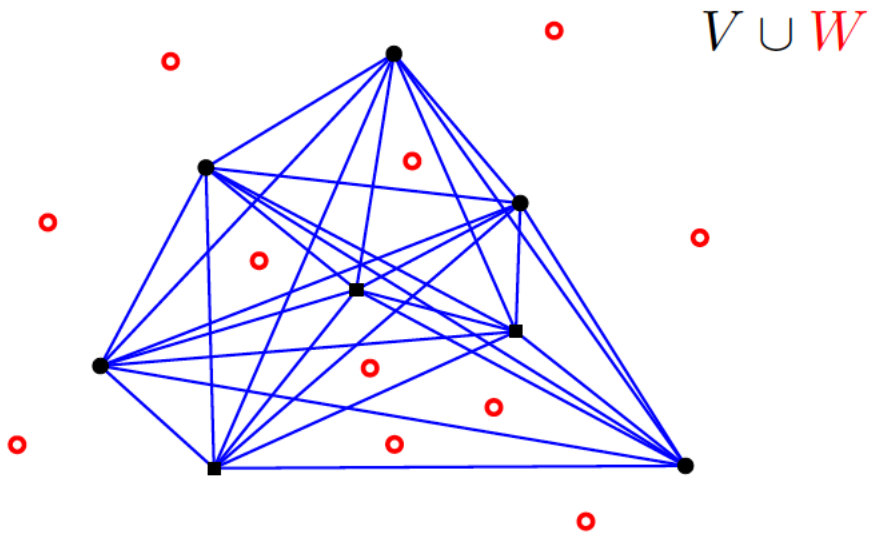

Figura 7 – Exemplo de gráfo não direcionado para V U W. (OLIVEIRA, 2009)

Para utilizar o algoritmos genéticos com o modelo m-PRC, o trabalho propõem dois modelos. O AGS (Algoritmo genético sequencial), que utiliza heurísticas GENIUS e 2-opt balanceada para ajustes finais para tentar melhor a solucão. E o AGH(Algoritmos genéticos H-1-PRC), que utiliza heurísticas H-1-PRC-MOD e 2-opt balanceada em todo o processo de resolução.

Pode-se concluir que a utilização de algoritmos geneticos para a resolução de de uma adaptação do problema de rotas de cobertura de veículos como bastante relevantes e de fácil manipulação. O modelo AGS resolve o problema de forma rápida e tem uma fácil implementação dentro dos critérios de comparação adotadas. O modelo AGH é mais lento e não conseguiu encontrar a solução para alguns exemplos.

2.4 Algorítimos genéticos para busca de caminhos

Buscando melhorar a eficiência dos algoritmos de busca, foram criadas formas híbridas, onde utilizam inteligência artificial para analisar todo o percusso e ajudar o algoritmo de busca a decidir 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 módulo para calcular os sub-caminhos ao longo do processo de busca, cada vez que o módulo é chamado, herda informações da chamada anterior, tendo um

ganha considerado de desempenho. O módulo é chamado, quando nenhum dos valores dados pelo BFS são melhores que o valor atual. A Figura 8 mostra uma fluxograma do funcionamento do modelo.

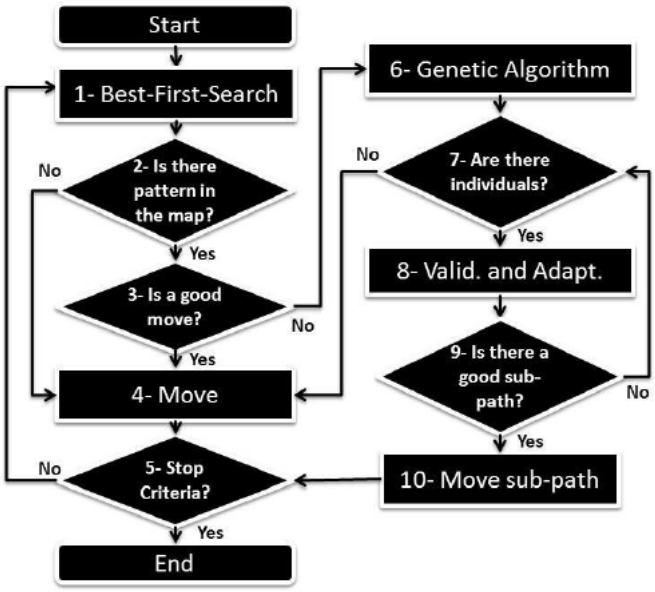

Figura 8 – Fluxograma do PPGA (SANTOS, 2012)

1. **Best-First-Seach** - Seleciona o nó atual e seleciona o menor valor na lista de nós abertos.

2. **Existe um padrão?** - No modulo de AG existe uma contagem de proporção de vezes que sub-caminho são utilizados pelas tentativas de utilização, se essa proporção for baixa, é definido que não existe um padrão na busca, então, não utiliza o modulo de AG.

3. **É um bom movimento?** - Verifica se o nó encontrado tem o valor menor que o atual, isto é, se aproxima mais do destino.

4. **Movimento** - Adiciona o nó atual para a lista fechada.

5. **Critério de parada?** - Se o ó atual é o nó de destino, o algoritmo termina, se não, continua para a próxima iteração.

6. **Algoritmo genético** - É obtida apenas uma geração com os seguintes critérios:

   a) A população inicial tem 4 indivíduos.

   b) Gera os primeiros descendentes utilizando com 4 primeiros indivíduos com um critério de 50%.

   c) Gera os próximos descendentes utilizando, pela mutação da população inicial com um critério de 25%.

d) É calculado a aptidão para 12 indivíduos. O elemento utilizado na ultima geração recebe o valor valor.

7. **Existem indivíduos?** - Verifica se existem indivíduos para serem avaliados na população. Se nenhum dos indivíduos forem validos, o algoritmo continua a manutenção do nó atual.

8. **Validação e adaptação** - Ajuda a garantir ageração e adaptação de bons indivíduos.

9. **Existe um bom sub-caminho?** - Os indivíduos gerados são testados, um em cada iteração, se não for um bom elemento, 'removido da lista da população.

10. **Movimento para o sub-caminho** - O elemento será usado como um sub-caminho composto pelos nós que serão adicionados à lista fechada. O nó real se tornar o último nó do sub-caminho. Este elemento será inserido para a próxima geração do GA.

A Figura 9 mostra o resultado de caminho percorrido pelo BFS em um ambiente simulado. Na Figura 10 mostra o caminho do PPGA esta utilizando o mesmo ambiente da Figura 9.

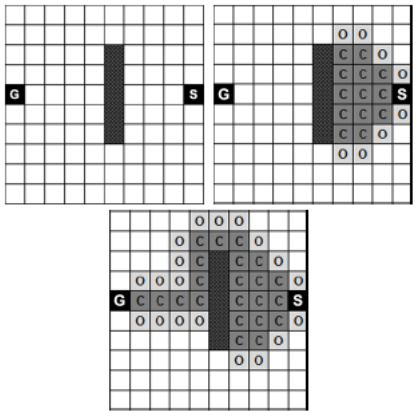

Figura 9 – Caminho criado pelo BFS (SANTOS, 2012)

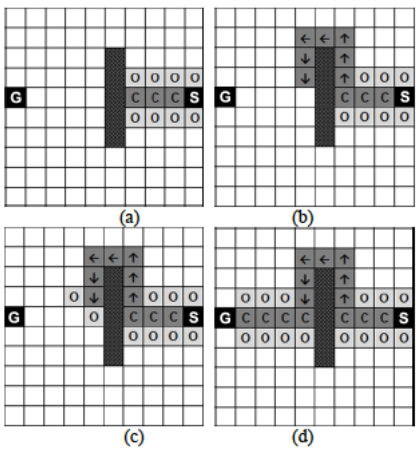

Figura 10 – Caminho criado pelo PPGA (SANTOS, 2012)

Os resultados da análise 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).

2.5 Algoritmos 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 número 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 paralelização 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). A Figura 11 mostra o desenho do modelo proposto.

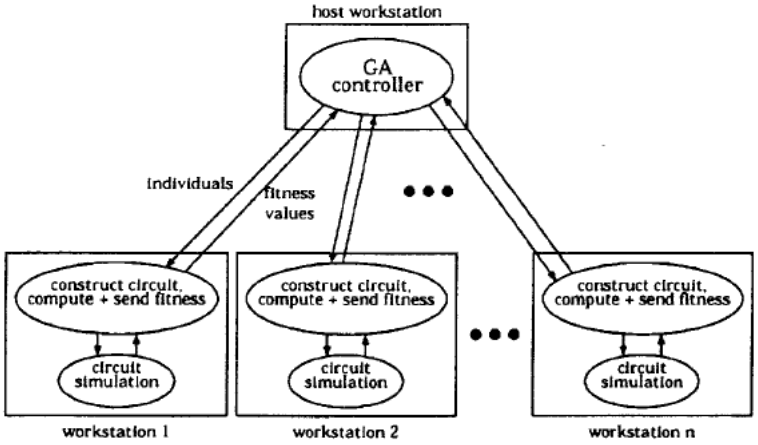

Figura 11 – Desenho do modelo de (D.LOHN SILVANO P. COLOMBANO, 2000)

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 cálculo 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. A Figura 12 mostra o desenho do modelo.

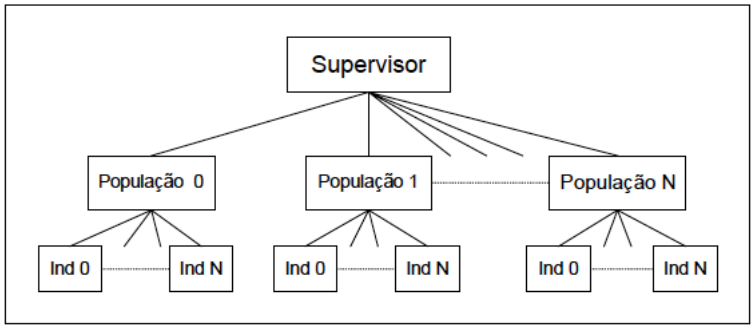

Figura 12 – Protótipo de (MOLE, 2002)

3 Proposta

A proposta deste trabalho é um modelo de busca de caminho do algoritmo A* com IA em uma arquitetura paralela. O A* tem sua utilização muito popular no ramo de jogos eletrônicos e existe uma necessidade na melhoria do desempenho destes algoritmos nesse meio. (GRAHAM, 2003)

A primeira parte do modelo que iremos propor, trata-se da utilização de inteligência artificial, em especifico, algoritmos genéticos. Uma arquitetura híbrida foi demonstrada indicando sucesso na melhoria de desempenho. (LEIGH; MILES, 2007)

Utilizar algoritmos genéticos de forma paralela mostra uma grande melhoria de desempenho conforme o numero de *threads* é aumentado. (ROSHANI, 2015)

Para a utilização em jogos eletrônicos é preciso que o modelo execute em tempo real, recalculando trajeto a cada iteração gráfica, tentando a partir daí, encontrar a uma solução mais próxima da ótima. Uma forma em tempo real do A* utilizando algoritmos genéticos teve resultados positivos. (MACHADO et al., 2011)

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 4 vezes nas páginas 6, 20, 21 e 22.

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 8.

BOTEA, M. M. A. Near optimal hierarchical path-finding. *Journal of Game Development, Vol. 1*, p. 7–28, 2004. Citado 2 vezes nas páginas 4 e 7.

CAIN, T. *Practical Optimizations for A**. [S.l.: s.n.], 2002. Citado 2 vezes nas páginas 4 e 10.

CHAMPANDARD, J. N. e A. J. *The Secrets of Parallel Pathfinding on Modern Computer Hardware*. 2010. Disponível em: <https://software.intel.com/en-us/articles/the-secrets-of-parallel-pathfinding-on-modern-computer-hardware>. Acesso em: 29/05/2016. Citado na página 12.

CHANG, J. *Making the Shortest Path Even Quicker*. 2009. Disponível em: <http://research.microsoft.com/en-us/news/features/shortestpath-070709.aspx>. Acesso em: 29/05/2016. Citado na página 12.

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 20, 21 e 22.

GRAHAM, H. M. e. S. S. R. Pathfinding in computer games. *The ITB Journal*, v. 4, 2003. Citado 2 vezes nas páginas 9 e 23.

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 7 e 8.

HOLLAND, J. *Adaptation in Natural and Artificial Systems*. Ann Arbor, MI, USA: University of Michigan Press, 1975. Disponível em: <http://books.google.com/books?id=YE5RAAAAMAAJ>. Citado na página 13.

KARP, R. Reducibility among combinatorial problems. In: MILLER, R.; THATCHER, J. (Ed.). *Complexity of Computer Computations*. [S.l.]: Plenum Press, 1972. p. 85–103. Citado na página 5.

KORF, R. E. Linear-space best-first search. *Artificial Intelligence*, v. 62, n. 1, p. 41 – 78, 1993. ISSN 0004-3702. Disponível em: <http://www.sciencedirect.com/science/article/pii/000437029390045D>. Citado 3 vezes nas páginas 4, 5 e 7.

KORF, R. E. Abstraction, reformulation, and approximation: 4th international symposium, sara 2000 horseshoe bay, usa, july 26–29, 2000 proceedings. In: _____. Berlin, Heidelberg: Springer Berlin Heidelberg, 2000. cap. Recent Progress in the Design and Analysis of Admissible Heuristic Functions, p. 45–55. ISBN 978-3-540-44914-0. Disponível em: <http://dx.doi.org/10.1007/3-540-44914-0_3>. Citado na página 9.

LEIGH, S. h. J. L. R.; MILES, C. Using a genetic algorithm to explore a*-like pathfinding algorithms. 2007. Citado na página 23.

LUCAS, D. C. Algoritmos genéticos: uma introdução. Universidade Federal do Rio Grande do Sul, 2002. Disponível em: <http://www.inf.ufrgs.br/~alvares/INF01048IA/ApostilaAlgoritmosGeneticos.pdf>. Citado 2 vezes nas páginas 13 e 15.

MACHADO, A. F. da V. et al. Real time pathfinding with genetic algorithm. In: *2011 Brazilian Symposium on Games and Digital Entertainment, Salvador, Bahia, Brazil, November 7-9, 2011*. [s.n.], 2011. p. 215–221. Disponível em: <http://dx.doi.org/10.1109/SBGAMES.2011.23>. Citado 2 vezes nas páginas 9 e 23.

MILLER, W. Applying parallel programming to path-finding with the a* algorithm. 2010. Citado 2 vezes nas páginas 4 e 12.

MOLE, V. L. D. Algoritmos genéticos – uma abordagem paralela baseada em populações cooperantes. 2002. Citado 3 vezes nas páginas 20, 21 e 22.

MUHLENBEIN, H. Evolution in time and space - the parallel genetic algorithm. 2000. Citado 3 vezes nas páginas 20, 21 e 22.

OLIVEIRA, W. A. de. Algoritmo genético para o problema de rotas de cobertura multiveículo. 2009. Citado na página 17.

PATEL, A. *A*'s Use of the Heuristic*. 2010. Disponível em: <http://theory.stanford.edu/~amitp/GameProgramming/Heuristics.html>. Acesso em: 29/05/2016. Citado 2 vezes nas páginas 9 e 10.

POLLACK, W. W. M. Solutions of the shortest-route problem-a review. *Operations Research*, INFORMS, v. 8, n. 2, p. 224–230, 1960. ISSN 0030364X, 15265463. Disponível em: <http://www.jstor.org/stable/167205>. Citado na página 4.

PONTEVIA, P. Pathfinding is not a star. *Autodesk, white paper*, p. 1–6, 2008. Citado 2 vezes nas páginas 4 e 11.

ROSHANI, M. K. S. R. Parallel genetic algorithm for shortest path routing problem with collaborative neighbors. 2015. Citado na página 23.

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 7 vezes nas páginas 4, 6, 9, 11, 18, 19 e 20.

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 11.

WANG, K. hsin C.; BOTEA, A. Fast and memory-efficient multi-agent pathfinding. In: *In ICAPS*. [S.l.: s.n.], 2008. p. 380–387. Citado na página 11.