INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DO SUDESTE DE MINAS GERAIS CÂMPUS RIO POMBA

ULYSSES OZÓRIO SANTOS

BEST-FIRST SEARCH COM ALGORITMO GENÉTICO PARA OTIMIZAÇÃO DE ESPAÇO EM PROBLEMAS DE BUSCA DE CAMINHOS

Rio Pomba

2012

ULYSSES OZÓRIO SANTOS

BEST-FIRST SEARCH COM ALGORITMO GENÉTICO PARA OTIMIZAÇÃO DE ESPAÇO EM PROBLEMAS DE BUSCA DE CAMINHOS

Trabalho de Conclusão de Curso apresentado ao Instituto Federal de Educação Ciência e Tecnologia do Sudeste de Minas Gerais – Câmpus Rio Pomba, como requisito parcial para a conclusão do Curso de Graduação em Ciência da Computação.

Orientador: Alex Fernandes da Veiga Machado

Rio Pomba

2012

FOLHA DE APROVAÇÃO

OZÓRIO SANTOS Ulysses. ***Best–First Search com Algoritmo Genético Para Otimização de Espaço em Problemas de Busca de Caminhos***. Trabalho de Conclusão de Curso, apresentado como requisito parcial à conclusão do curso Graduação em Ciência da Computação, do Instituto Federal de Educação Ciência e Tecnologia do Sudeste de Minas Gerais – Câmpus Rio Pomba, realizada no 2° semestre de 2012.

BANCA EXAMINADORA

_____

Prof. Dr. Alex Fernandes da Veiga Machado
Orientador

_____

Prof. Msc. Frederico de Miranda Coelho
Membro convidado 1

_____

Prof. Dr. Alessandra Martins Coelho
Membro convidado 2

Examinado (a) em: ____/____/_____.

Dedico este trabalho aos meus pais,
Cida e Jorge, por todo apoio e
incentivo.

AGRADECIMENTOS

Agradeço a Deus, por ter me dado forças para que conseguisse seguir em frente em minha jornada.

Agradeço à da minha família, especialmente aos meus pais, Cida e Jorge, que não pouparam esforços para que eu conseguisse concluir esta etapa, ao meu irmão Herman e ao meu primo Robson por estarem sempre me apoiando.

Agradeço minha namorada Renata pela paciência e todo apoio muito importantes para que eu conseguisse chegar ao final desta jornada.

Aos professores, que me proporcionado o conhecimento e ao Google, que de forma complementar, contribuiu para minha formação. À professora Alessandra que me ajudou, e muito, a concluir este trabalho.

A todas amizades que surgiram durante esse período.

E a todos aqueles que oraram e torceram positivamente por mim.

“Durma na escola, como todo mundo.” Charlie Harper - Dois Homens e Meio.

RESUMO

SANTOS, Ulysses Ozório. ***Best–First Search* com Algoritmo Genético Para Otimização de Espaço em Problemas de Busca de Caminhos**. Trabalho de Conclusão de Curso (Graduação em Ciência da Computação). Instituto Federal de Educação Ciência e Tecnologia do Sudeste de Minas Gerais – Câmpus Rio Pomba, Rio Pomba, 2012.

A busca de caminhos entre dois pontos é um problema muito estudado na área de jogos eletrônicos. Para aplicações multiagentes muitos algoritmos clássicos tornam-se inviáveis de se utilizar devido ao consumo de memória. Este trabalho apresenta uma modificação para o algoritmo clássico *Best-First Search* (BFS) que visa otimizar o seu consumo de memória. Esta modificação consiste em inserir um módulo composto pelo Algoritmo Genético para detecção de padrões de obstáculos do ambiente de busca. O algoritmo proposto é chamado de *Pattern based Pathfinding with Genetic Algorithm* (PPGA). Foi demonstrado que o PPGA obteve um desempenho melhor que o tradicional BFS em ambientes com obstáculos padronizados.

Palavras-chave: Busca de caminhos. Jogos eletrônicos. Algoritmo Genético. Metaheuristica. Reconhecimento de padrões.

ABSTRACT

This paper presents a novel method to optimize the process of finding paths using a model based on Genetic Algorithms and Best-First-Search (BFS) for multi - agents systems, such as video games and virtual reality environments. The proposed solution uses obstacle pattern detection based at online training system to guarantee the memory economy. The architecture named Patterned based Pathfinding with Genetic Algorithm (PPGA) uses a learning technique in order to create an agent adapted to the environment that is able to optimize the search for paths even in the presence of obstacles. We demonstrate that the PPGA architecture performs better than classic Best-First-Search algorithm in patterned environment.

Keywords: Pathfinding. Eletronic Games. Genetic Algorithm. Metaheuristic. Pattern Recognition.

LISTA DE TABELAS

Tabela 1 - Codificação do indivíduo

Tabela 2 - Configuração dos indivíduos

Tabela 3 - Configuração dos indivíduos com desvio

Tabela 4 - Configuração dos indivíduos com dois desvios

Tabela 5 - Codificação da solução parcial da Figura 7(a)

Tabela 6 - Codificação da solução parcial da Figura 7(b)

Tabela 7 - Resultados dos testes

LISTA DE FIGURAS

Figura 1 - Exemplo da execução do algoritmo BFS

Figura 2 - Dois exemplos do PPGA

Figura 3 - Movimentos em linha reta

Figura 4 - Movimentos com desvio

Figura 5 - Movimentos com dois desvios

Figura 6 - Validação das soluções parciais: (a) é válida, (b) será adaptada, (c) não é válida

Figura 7 - Adaptação da solução parcial

LISTA DE FLUXOGRAMAS

Fluxograma 1 - Módulo principal do PPGA

Fluxograma 2 - Módulo AG

LISTA DE GRÁFICOS

Gráfico 1 - Gráfico comparativo

LISTA DE SIGLAS

AG	Algoritmo Genético.
BFS	*Best–First Search.*
RTS	*Real Time Strategy.*
RPG	*Role Playing Game.*
PPGA	*Patterned based Pathfinding with Genetic Algorithm.*
RTP-GA	*Real Time Pathfinding with Genetic Agorithm,*
IDA*	*Iterative Depth A*.*
TRA*	*Triangulation Reduction A*.*
RBFS	*Recursive Best–First Search*
GAMMA	*Genetic Algorithm Manufactured Maneuvering Algorithm.*

SUMÁRIO

1 INTRODUÇÃO

1.1 OBJETIVO GERAL

1.2 OBJETIVOS ESPECÍFICOS

1.3 ORGANIZAÇÃO DO TRABALHO

2. TRABALHOS RELACIONADOS

3. *PATTERNED BASED PATHFINDING WITH GENETIC ALGORITHM*

3.1 BEST-FIRST SEARCH (BFS)

3.2 ALGORITMO GENÉTICO

3.3 O ALGORITMO PROPOSTO

3.3.1 Soluções Parciais

3.3.2 Representação do Movimento

3.3.3 Restrições

3.3.4 Função de Aptidão (*Fitness*)

3.3.5 Validação

3.3.6 Adaptação

3.3.7 Identificação de Ambientes sem Padrão

3.3.8 Fluxograma

4. EXPERIMENTOS

4.1 TECNOLOGIA

4.2 TESTES

4.3 RESULTADOS E ANÁLISE

5. CONCLUSÃO E TRABALHOS FUTUROS

PUBLICAÇÕES.....

REFERÊNCIAS

APÊNDICE A. - PSEUDOCÓDIGO........

1 INTRODUÇÃO

O problema de busca de caminhos é um problema básico em jogos eletrônicos, que consiste em encontrar um caminho válido de um ponto ao outro, no qual um personagem irá percorrer. Geralmente o ambiente onde ocorre a busca é representado por um grafo, composto por nós. Para cada nó do grafo é gerado um custo, que é determinado por uma heurística, dependendo do algoritmo utilizado (MATTHEWS, 2002).

Para esse tipo de problema alguns algoritmos clássicos, como o *Best-First Search* (BFS) (RUSSEL, 2004) e o A* (HART, 1968; RUSSEL, 2004) podem ser aplicados. Ambos são algoritmos determinísticos e utilizam heurísticas semelhantes, sendo que o primeiro não encontra, necessariamente, o menor caminho (RUSSEL, 2004) e o segundo já consegue encontrar essa solução ótima (HART, 1968; RUSSEL, 2004).

Em aplicações multiagentes, como jogos dos gêneros *Role Playing Game* (RPG) e *Real Time Strategy* (RTS), nos quais muitos personagens devem calcular rotas simultaneamente. O problema de busca de caminhos se torna um problema do tipo PSPACE - difícil (HOPCROFT, 1984; REIF, 1979), pois o número de nós armazenados cresce exponencialmente de acordo com a quantidade de agentes. Os algoritmos clássicos supracitados podem não apresentar um bom desempenho, devido ao consumo de memória (CAIN, 2002; HIGGINS, 2002). Alguns trabalhos, como o algoritmo *Iteractive Deeping* A* (IDA*) desenvolvido em Korf (1985), visam otimizar esse consumo de memória.

O uso de metaheurísticas para a adaptação ao ambiente é um tema abordado por diversos autores. São estratégias de alto nível que guiam os processos de pesquisa, e tem como objetivo explorar, de forma eficiente, o espaço de busca, a fim de encontrar soluções ideais tanto para problemas

simples, quanto para complexos. Estas são técnicas não determinísticas e podem utilizar heurísticas para guiar seu processo de pesquisa (BRUM e ROLI, 2003). São exemplos de metaheurísticas, Algoritmos Genéticos (AG), Busca-Tabu, Colônia de Formigas, GRASP, *Simulated Annealing*, dentre outras (BRUM e ROLI, 2003).

1.1 OBJETIVO GERAL

Otimizar o consumo de memória exigido pela busca de caminhos, utilizando a metaheurística AG combinada com o algoritmo determinístico BFS.

1.2 OBJETIVOS ESPECÍFICOS

- Identificar padrão dos obstáculos;
- Modelar as soluções a fim de permitir flexibilidade e adaptação ao ambiente;
- Utilizar treinamento *on–line* visando de torná-lo mais utilizável.
- Comparar e analisar os resultados obtidos na execução dos algoritmos clássicos A* e BFS com os do algoritmo proposto, em três ambientes diferentes (sem padrões, com padrões e misto).

1.3 ORGANIZAÇÃO DO TRABALHO

Este trabalho está organizado da seguinte forma: o capítulo 2 apresenta uma pesquisa sobre trabalhos relacionados voltados para o problema de busca de caminhos. As especificações do algoritmo proposto (PPGA) são mostradas no capítulo 3. Os testes e análise comparativa entre o PPGA, BFS e o algoritmo

A* são apresentados no capítulo 4. Finalmente, no capítulo 5, apresenta-se a conclusão e proposta de trabalhos futuros.

2. TRABALHOS RELACIONADOS

Os problemas de busca de caminhos é um problema muito estudado, onde o fator de complexidade espacial (consumo de memória) é muito visado. Os trabalhos de Korf (1985) e Björnsson et al (2005) propõem otimizar esse fator.

O algoritmo *Iterative Deepening A** (IDA*), proposto por Korf (1985), utiliza a busca em profundidade combinada com a heurística do A*. O resultado é um algoritmo que encontra a solução ótima, assim como o A*, com o gasto mínimo de memória. A desvantagem dessa técnica é que a árvore gerada na busca a cada iteração, deve ser gerada novamente. Com isso, o algoritmo exige muito processamento, o que pode torná-lo mais lento que o A*.

A proposta apresentada por Björnsson et al (2005), *Fringe Search,* visa otimizar o algoritmo IDA*. Esta otimização, tem como foco eliminar a principal deficiência do IDA* citada acima. Com isso, a cada iteração, não é mais necessário gerar a árvore toda novamente, pois o algoritmo armazena os nós folhas da mesma, o que faz que, na próxima iteração, a busca se inicie a partir desses nós. Com isso, o algoritmo *Fringe Search* consegue ser mais rápido que o IDA*.

O *Recursive Best–First Search (RBFS*), proposto por Korf (1992), é um algoritmo que trabalha com a redução do consumo de memória para o processo de busca de caminhos. Este algoritmo utiliza a mesma heurística do BFS e promove uma otimização espacial para o mesmo.

A técnica de reconhecimento de padrão de obstáculos do ambiente é um assunto tratado por Demyen e Buro (2006). O algoritmo por eles proposto, chamado de *Triangulation Reduction A** (TRA*), utiliza uma abstração do ambiente a fim de determinar o padrão dos obstáculos do ambiente. Este visa otimizar o tempo gasto no processamento do algoritmo A*. Esse método tem como desvantagem, requisitar um formato específico dos obstáculos do

ambiente, que devem ser poligonais, o que pode inviabilizar sua utilização em qualquer tipo de ambiente.

A utilização do AG, em conjunto com métodos heurísticos de busca, é um assunto pouco estudado. Contudo, alguns métodos podem ser citados. O algoritmo *GA-Manufactured Maneuvering Algorithm* (GAMMA), proposto por Leigh et al (2007), utiliza o AG a fim de promover uma otimização tempo especial para o algoritmo A*. O AG é utilizado para modificar o comportamento do algoritmo A*, determinando o melhor comportamento para obter um melhor desempenho no ambiente. Essa proposta necessita de um treinamento *off-line*, e consegue atingir um desempenho até 1271% mais rápido que o tradicional A*.

O trabalho de Burchardt e Salomon (2006) propõe uma busca de caminho utilizando somente o AG. A função de avaliação utilizada se baseia na distância percorrida pelo robô e nas suas colisões com obstáculos do ambiente.

3. *PATTERNED BASED PATHFINDING WITH GENETIC ALGORITHM*

O algoritmo *Patterned based Pathfinding with Genetic Algorithm* (PPGA) usa como base o clássico BFS com sua heurística gulosa, no qual foi introduzido um módulo que usa o AG para gerar sub-caminhos que serão utilizados para contornar obstáculos, a fim de obter uma redução dos nós armazenados nas listas aberta e fechada. Em suma, esta modificação tem a finalidade de obter eficiência no consumo de memória durante o processo de busca.

Antes de descrever e explicar o funcionamento do PPGA apresenta-se, resumidamente o BFS e o AG.

3.1 *BEST-FIRST SEARCH* (BFS)

O *Best–First Search* (BFS) é um algoritmo de busca que utiliza uma função heurística F(n) que é dada por (1):

$$F(n) = h(n) \tag{1}$$

em que:

h(n) é a distância entre o nó corrente n e o nó objetivo.

Durante a busca, os nós avaliados são armazenados em duas estruturas chamadas de lista aberta e lista fechada. A cada iteração, um novo nó corrente é escolhido por meio da função heurística. O nó que tem o menor valor será selecionado. Enquanto este é armazenado na lista fechada, seus nós vizinhos serão armazenados na lista aberta. O processo será repetido até que o nó corrente seja o nó objetivo.

Este método é considerado um método guloso, pois visa sempre melhorar o seu estado, escolhendo sempre o nó que está mais próximo do nó

objetivo para ser o nó corrente. Por esse motivo ele encontra o caminho de forma mais eficiente, sem, no entanto, sempre encontrar o menor caminho.

Pode-se observar, na Figura 1, a quantidade de nós armazenados nas listas aberta “O“ e fechada “C”.

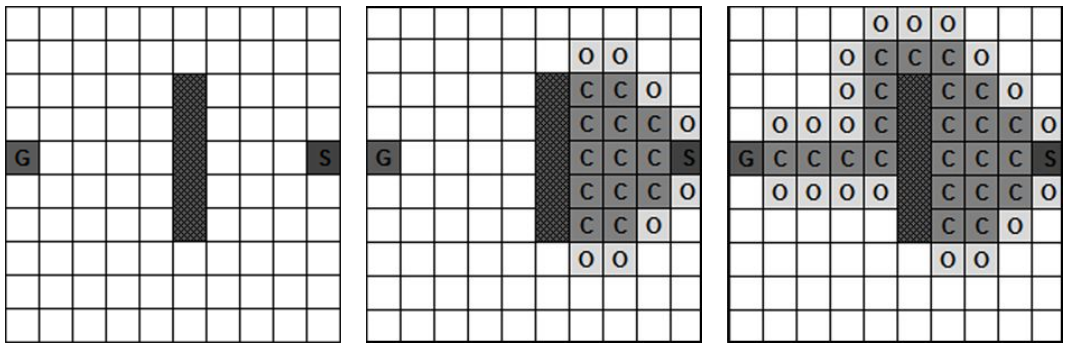


Figura 1 - Exemplo da execução do algoritmo BFS

Fonte: Santos et al (2012b)

Para maiores esclarecimentos sobre o algoritmo BFS, veja (RUSSELL, 2004).

3.2 ALGORITMO GENÉTICO

O Algoritmo Genético (AG) é uma metaheurística inspirada na natureza. Baseia–se na teoria de C. Darwin, onde propõe que em uma população de indivíduos aquele que for mais apto terá maior probabilidade de reprodução e com isso tem mais chance de permutar suas características genéticas para os indivíduos das próximas gerações (SOARES, 1997).

Esse conceito de evolução é representado computacionalmente com o objetivo de buscar melhores soluções para um determinado problema. Esta busca se da através da evolução de populações de soluções, que são codificadas e representadas por cromossomos artificiais (SOARES, 1997).

O AG é na maioria das vezes aplicado a problemas de otimizações complexos, onde é necessário combinar vários parâmetros para se obter uma

melhor solução, pode também ser aplicado a problemas que não pode ser representados matematicamente e problemas que apresentam um grande espaço de busca (SOARES, 1997). Este pertence à classe dos métodos probabilísticos de busca e otimização, porém não é aleatório, pois tenta dirigir a busca para regiões onde possam ser encontradas as melhoras soluções (TANOMARU, 1995).

Cada possível solução que compõe a população é representada por um cromossomo ou indivíduo, este é representado por uma sequência de símbolos gerados a partir de um alfabeto finito. Tanto a representação quanto o alfabeto depende de cada problema (TANOMARU, 1995).

A população é formada por operadores genéticos, tais como, cruzamento, mutação e inversão. Não existe uma definição muito rígida em relação aos algoritmos genéticos, porém estes, na maioria dos casos, têm em comum os elementos: população de indivíduos, seleção (de acordo com a aptidão), cruzamento e mutação (MITCHELL 1996).

A população inicial é gerada aleatoriamente na primeira iteração do algoritmo, mas, para as próximas iterações, utilizam-se os melhores indivíduos provenientes da iteração anterior (MITCHELL, 1996).

O operador de cruzamento é responsável pela troca de material genético entre dois indivíduos pais, gerando dois indivíduos filhos. O operador de mutação modifica genes aleatoriamente.

Para seleção natural é usada uma função de aptidão (*Fitness*). Essa função gera um valor que é uma medida de quão adaptada ao ambiente o indivíduo está (TONOMARU, 1995). De acordo com o esse valor o(s) indivíduo(s) mais apto(s) é(são) selecionado(s) (MITCHELL 1996).

Para maiores esclarecimentos sobre Algoritmo Genéticos, veja (SOARES, 1997; TONOMARU, 1995).

3.3 O ALGORITMO PROPOSTO

A cada iteração do BFS é avaliado se foi atingido o critério para a utilização do módulo AG. Quando isso ocorre o AG itera sobre uma população de soluções parciais candidatas e, por meio de um critério elitista, é selecionada uma solução parcial para que seja utilizada no ambiente. Após a utilização o algoritmo retorna ao processamento do BFS.

Quando o módulo AG é ativado, ele utiliza, quando há, a solução parcial, usada anteriormente, como herança para gerar uma nova população melhorada. As soluções são selecionadas via critério elitista, utilizando uma função de aptidão (FA) e o processo de validação.

A solução parcial retornada pelo módulo AG, quando aplicada ao ambiente, é convertida em um caminho que é composto por nós. Estes são adicionados à lista fechada do BFS, sem a necessidade de avaliar sua vizinhança, proporcionando uma redução dos nós armazenados durante a busca. Na Figura 2 são mostrados exemplos da utilização das soluções parciais no ambiente. As setas representam as soluções parciais geradas pelo AG e aplicados ao ambiente.

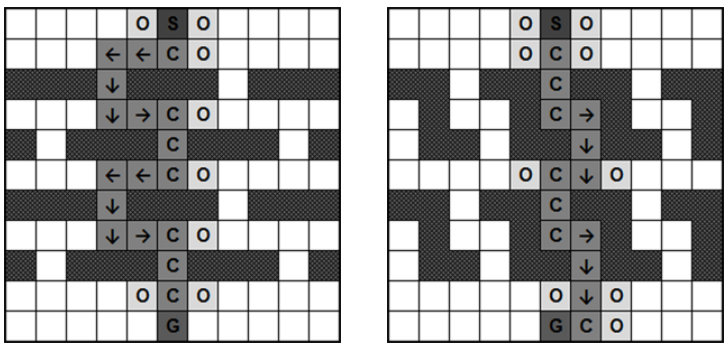


Figura 2 - Dois exemplos do PPGA
Fonte: Santos et al (2012b)

3.3.1 **Soluções Parciais**

As soluções parciais são caminhos alternativos utilizados para contornar obstáculos presentes no ambiente. Para se ter este comportamento, as

soluções foram codificadas a fim de se obter maior flexibilidade (cada indivíduo foi convertido em um caminho composto por nós). Assim, as soluções parciais foram modeladas contendo quatro vetores de movimento, dois verticais e dois horizontais.

Nas soluções parciais, os indivíduos são compostos por oito genes representados por números inteiros. Os quatro primeiros representam o tamanho e os quatro últimos as direções e os sentidos dos vetores de movimento. A Tabela 1 apresenta, com mais detalhes, a codificação dos indivíduos.

Tabela 1 - Codificação do indivíduo

Tipo	Intervalo	Descrição
Inteiro	1 até altura do mapa	Distância do Movimento 1 (DM1)
Inteiro	1 até largura do mapa	Distância do Movimento 2 (DM2)
Inteiro	1 até altura do mapa	Distância do Movimento 3 (DM3)
Inteiro	1 até largura do mapa	Distância do Movimento 4 (DM4)
Inteiro	-1 até 1	Direção do Movimento 1 (M1)
Inteiro	-1 até 1	Direção do Movimento 2 (M2)
Inteiro	-1 até 1	Direção do Movimento 3 (M3)
Inteiro	-1 até 1	Direção do Movimento 4 (M4)

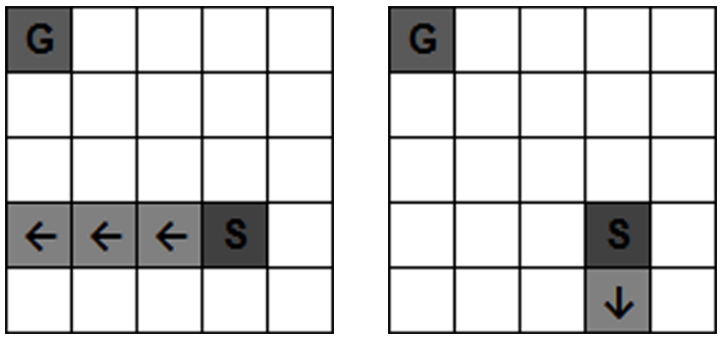
Os valores dos quatro primeiros genes (tamanho) podem variar de acordo com as dimensões do ambiente (altura, para vetores verticais e largura, para vetores horizontais). Os genes DM1 e DM3 representam vetores verticais. Já os genes DM2 e DM4, vetores horizontais.

Os valores dos quatro últimos genes (direção e sentido) podem variar de -1 a 1. Os genes M1 e M3 correspondem, respectivamente, aos genes DM1 e DM3. Do mesmo modo, M2 e M4 correspondem aos genes DM2 e DM4. Cada par de genes (DM e M) corresponde a um vetor de movimento. DM1 e M1, por exemplo, correspondem ao vetor vertical.

3.3.2 **Representação do Movimento**

Como já descrito anteriormente, as soluções parciais representam caminhos alternativos nos quais cada indivíduo é convertido em um caminho composto por nós. A Figura 3 visa demonstrar como os indivíduos, codificados na Tabela 2, são representados no ambiente. Considere “G” o nó objetivo e “S“ o nó de partida.

Os indivíduos representados na Tabela 2 (primeira e segunda linha da Tabela 2) têm apenas um vetor de movimento. No primeiro indivíduo, o vetor correspondente aos genes DM2 e M2, tem o sentido horizontal e seu tamanho igual a 3. Por isto, na Figura 3(a) a solução parcial aplicada ao ambiente é composta por 3 nós.



(a) (b)

Figura 3 - Movimentos em linha reta
Fonte: Santos et al (2012b)

Tabela 2 - Configuração dos indivíduos

DM 1	DM 2	DM 3	DM 4	M 1	M 2	M 3	M4
2	3	4	1	0	-1	0	0
1	3	2	1	-1	0	0	0

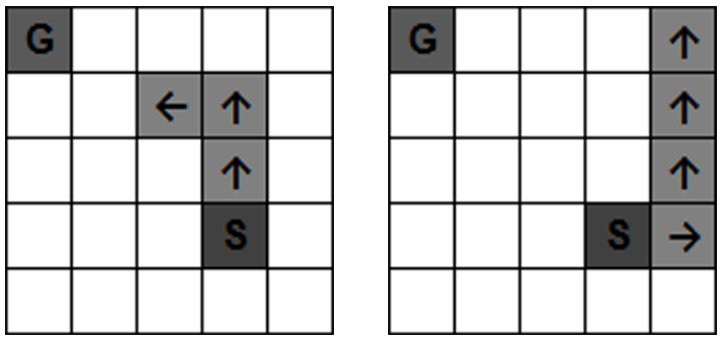
Para o segundo o vetor, correspondente aos genes DM1 e M1, tem-se o sentido vertical e tamanho 1, Figura 3(b).

Os vetores, os quais seus genes M tenham valores nulos (0) são vetores sem movimentos ou vetores nulos. Por exemplo, os vetores correspondentes à Tabela 2, na primeira linha, DM1 e M1, DM3 e M3 e o DM4 e M4.

Tabela 3 - Configuração dos indivíduos com desvio

DM 1	DM 2	DM 3	DM 4	M 1	M 2	M 3	M4
2	1	4	1	1	-1	0	0
1	1	3	1	0	1	1	0

A Tabela 3 apresenta mais dois indivíduos, com dois vetores de movimento cada. No primeiro indivíduo (primeira linha da Tabela 3), os genes DM1 e M1 correspondem a um vetor (sentido vertical) e os genes DM2 e M2 à outro (sentido horizontal). No segundo (segunda linha da Tabela 3) os genes DM2 e M2 é um vetor e os genes DM3 e M3 o outro vetor. Os indivíduos são representados na Figura 4.



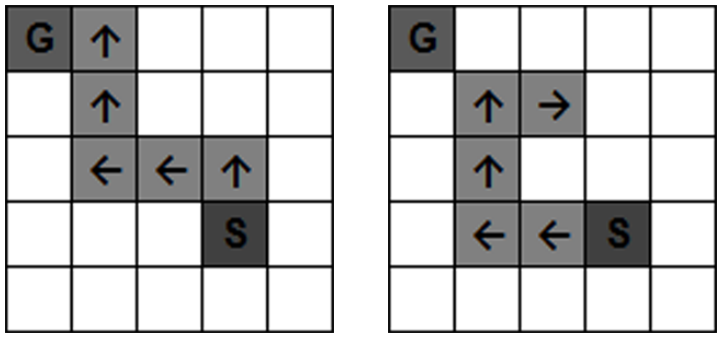
(a) (b)

Figura 4 - Movimentos com desvio
Fonte: Santos et al (2012b)

Na Tabela 4, os indivíduos apresentados possuem 3 vetores de movimento. Estes são representados, no ambiente, conforme ilustrado na Figura 5.

Tabela 4 - Configuração dos indivíduos com dois desvios

DM 1	DM 2	DM 3	DM 4	M 1	M 2	M 3	M4
1	2	2	1	1	-1	1	0
1	2	2	1	0	-1	1	1



(a) (b)

Figura 5 - Movimentos com dois desvios
Fonte: Santos et al (2012b)

3.3.3 **Restrições**

As soluções parciais geradas devem obedecer algumas restrições. Esta medida foi adotada para melhor aproveitamento das mesmas.

Existem dois tipos de restrições, as fixas e as dinâmicas. As restrições fixas independem do ambiente e do estado atual da busca. Seu objetivo é evitar duas situações: (1) não permitir a geração de soluções parciais, cujos valores dos quatro últimos genes sejam nulos (zero), devido a esse tipo de solução parcial não apresentar vetores de movimento; (2) evitar sobreposição de vetores de movimentos. Esse tipo de solução se caracteriza pela seguinte codificação: o valor de M2 é nulo e os genes M1 e M3 têm valores opostos (1 e -1 ou vice-versa), ou M3 com o valor nulo e os valores de M2 e M4 opostos.

As restrições dinâmicas se caracterizam por dependerem de informações do estado atual da busca. A cada iteração da busca do BFS, um nó é escolhido para ser o nó corrente. Este tem seus nós vizinhos, caso não sejam obstáculos, avaliados e alocados na lista aberta. Quando as soluções parciais são geradas, é levado em consideração o nó corrente. As soluções parciais, que apresentam o primeiro vetor não nulo, com a direção e sentido de um vizinho do nó corrente, que seja um obstáculo, devem ser evitadas. Caso um vizinho inferior do nó corrente for um obstáculo, as soluções parciais que

tenham seus genes M1 igual a -1 ou M1 e M2 iguais a zero e M3 igual a -1, não são geradas.

3.3.4 **Função de Aptidão (*Fitness*)**

A Função de Aptidão (FA) foi desenvolvida a fim de identificar as soluções parciais mais promissoras, visto que esta função não garante a seleção destas. Ela garante que as soluções que tenham o maior valor sejam as que atendam as seguintes condições:

- Estejam proporcionando um caminho que deixa o nó corrente da busca mais próximo do objetivo;
- Não estejam proporcionando um caminho cíclico;

A função foi definida por (2):

$$FA = TETO - MH - MT \cdot 0{,}1 \tag{2}$$

em que:

TETO é um valor estipulado para garantir que a função seja de maximização, ou seja, a melhor solução deverá ter o valor de FA mais alto. Este valor foi dado pela equação (3):

$$TETO = (LARGURA + ALTURA) * 3 \tag{3}$$

em que:

LARGURA e ALTURA se referem à largura e altura do ambiente.

MH é o valor da heurística utilizada para medir a distância entre o último nó, que compõe o caminho gerado pela solução parcial, até o nó objetivo. É responsável por garantir que a solução esteja deixando o nó corrente mais próximo do nó objetivo. Portanto, quanto maior o seu valor, pior será a solução. Este valor é dado por (4):

$$MH = |Gx - (Sx + (M2 * DM2 + M4 * DM4) \cdot LARGURA)| + |Gy - (Sy + (M1 * DM1 + M3 * DM3) \cdot ALTURA)| \quad (4)$$

em que:

Sx e Sy representam a posição do nó corrente no ambiente;

Gx e Gy representam a posição do nó objetivo.

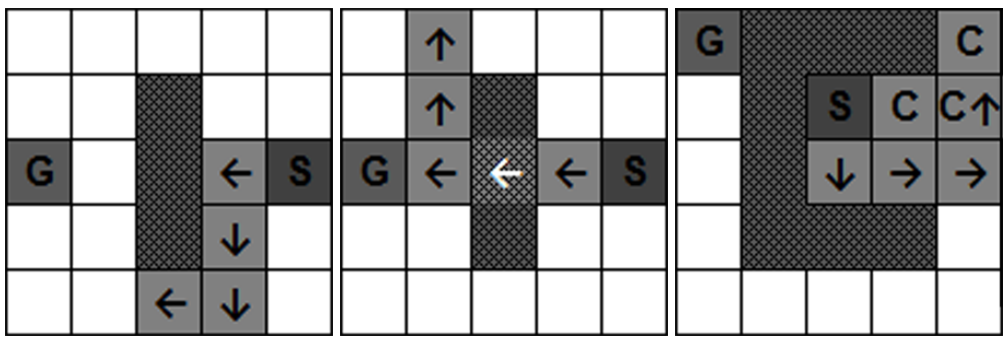
MT é o valor que determina o tamanho do caminho gerado pela solução parcial. Este, em conjunto com o valor MH, determina se o caminho é cíclico. Seu valor é dado por (5):

$$MT = (M1 \cdot DM1 + M3 \cdot DM3) \cdot ALTURA + (M2 \cdot DM2 + \cdot DM4) \cdot LARGURA \quad (5)$$

3.3.5 **Validação**

O processo de validação tem o propósito de efetivamente selecionar a solução parcial mais apta (válida) e eliminar a menos apta (inválida). Consiste de duas fases, a primeira avalia se a solução parcial atende os seguintes requisitos:

- O seu valor de MH é menor que o valor de F do nó corrente;
- Se o último nó que compõe o caminho gerado pela solução, é um ponto que já fora armazenado na lista fechado do BFS.



(a) (b) (c)

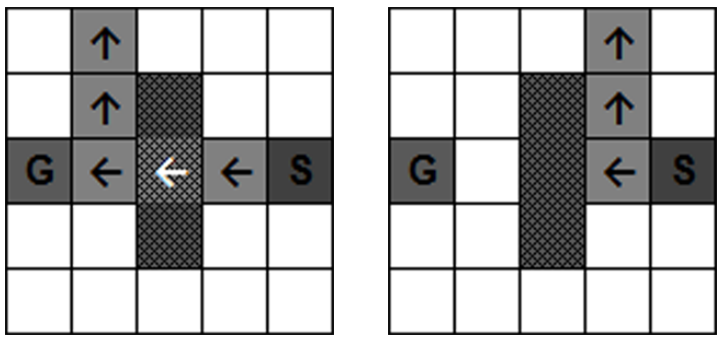
Figura 6 - Validação das soluções parciais: (a) é válida, (b) será adaptada, (c) não é válida
Fonte: Santos et al (2012b)

Quando esses requisitos não são atendidos, a solução parcial é considerada inválida. Atendendo esses requisitos a solução passa para segunda fase do processo. Na Figura 6(c), mostra uma solução que não atende o segundo requisito.

A segunda fase tem o objetivo de identificar se algum nó que compõe o caminho gerado pela solução passa sobre algum nó obstáculo (colisão), Figura 6(b). Quando identificado esse ocorrência a solução deve passar pelo processo de adaptação. No caso não ocorra nenhuma colisão a solução será aplicada no ambiente, Figura 6(a).

3.3.6 **Adaptação**

O processo de adaptação tem como objetivo a indução da adaptação no ambiente e um melhor aproveitamento das soluções geradas. Esse processo consiste em modificar a codificação da solução parcial, adaptando-a aos obstáculos presentes no ambiente. Após passar por esse processo é calculado um novo valor de FA para a solução.



(a) (b)

Figura 7 - Adaptação da solução parcial
Fonte: Santos et al (2012b)

Na Figura 7(a) é demonstrada uma ocorrência de colisão. Esse tipo de situação faz com que a solução seja adaptada. A Tabela 5 representa a codificação original da solução.

Tabela 5 - Codificação da solução parcial da Figura 7(a)

DM 1	DM 2	DM 3	DM 4	M 1	M 2	M 3	M 4
4	3	2	1	0	-1	1	0

Tabela 6 - Codificação da solução parcial da Figura 7(b)

DM 1	DM 2	DM 3	DM 4	M 1	M 2	M 3	M 4
4	1	2	1	0	-1	1	0

Após o processo de adaptação, a solução tem sua codificação alterada, conforme apresentado na Tabela 6. A Figura 7(b) demonstra como a solução adaptada é representada no ambiente.

3.3.7 **Identificação de Ambientes sem Padrão**

A fim de evitar o uso desnecessário do módulo AG, quando este não estiver sendo muito eficaz, foi implementado um mecanismo que o desabilita. Geralmente esta ineficácia ocorre quando o algoritmo é executado em ambientes onde não existe um padrão de obstáculos, com isso, o módulo AG não consegue gerar soluções parciais que se adaptem e, como consequência, o seu desempenho cai drasticamente.

Para identificar esses casos foram definidas duas métricas: a porcentagem da distância percorrida (D) dada pela equação (6) e a taxa de sucesso (S), apresentada em (7).

$$\frac{MDcorrente}{MDtotal} \quad D = \frac{MDcorrent}{MDtotal} \tag{6}$$

em que:

MDcorrente é a distância do nó corrente para o destino;

MDtotal é a distância total do nó corrente, do início da pesquisa, para o destino (utilização da heurística de Manhattan).

$$\frac{\Sigma SPsucesso}{\Sigma SPfalha} \qquad S = \frac{\sum SPsucess}{\sum SPfalha} \tag{7}$$

em que:

SPsucesso é o número de vezes que uma solução parcial foi utilizada no ambiente;

SPfalha é número de vezes que não foi possível utilizar a solução parcial. Isso ocorre quando o módulo AG não identifica nenhuma solução parcial válida.

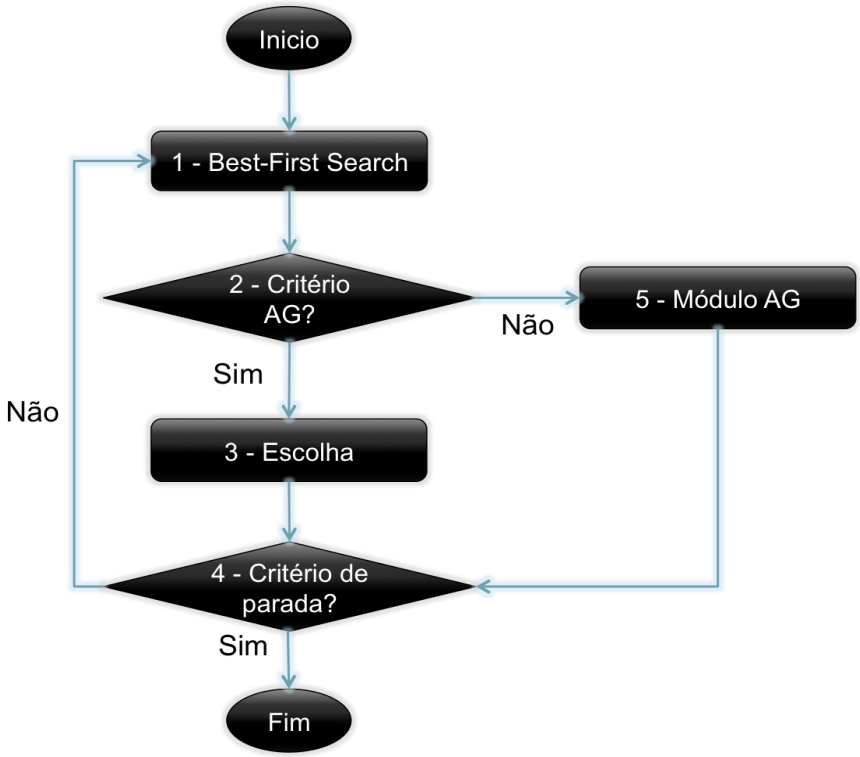
A fim de identificar se o local da busca apresenta algum padrão de obstáculos, foram definidos valores limites para D e S. Foram realizados testes preliminares e estabelecido que, se o valor de D fosse menor que 0,5 e S menor que 0,4, o ambiente não teria padrão, equivalendo ao valor zero (0), caso contrário, padrão receberia o valor um (1), conforme pode ser visto em (8)

$$Padrão = \begin{cases} 0, & D < 0{,}5 \text{ e } S < 0{,}4 \\ 1 \end{cases} \tag{8}$$

O valor de Padrão determina se o módulo AG será desabilitado ou não. Para Padrão igual a zero (0), o módulo AG é desabilitado, caso contrário, quer dizer que o módulo AG está sendo eficaz e, portanto, deve continuar habilitado.

3.3.8 **Fluxograma**

O fluxograma, a seguir (Fluxograma 1), mostra o funcionamento detalhado do módulo principal do PPGA (BFS).

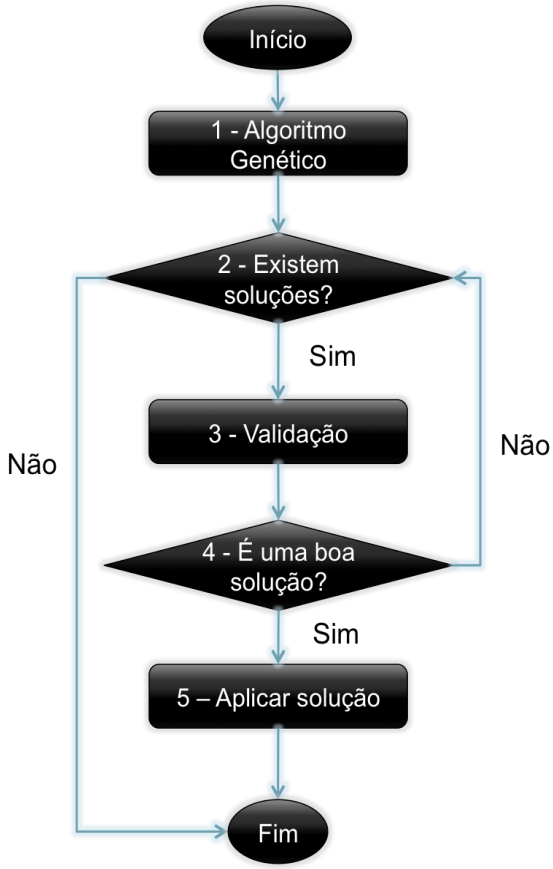


Fluxograma 1 - Módulo principal do PPGA

1. *Best-First Search* – A cada iteração do algoritmo, o nó corrente é adicionado à lista fechada e é avaliada a sua vizinhança. Para cada um dos nós válidos (não obstáculos) presente na vizinhança é gerado o valor de F, e estes são adicionados à lista aberta. Após isso o indivíduo que apresenta o menor valor de F, será escolhido para ser o novo nó corrente.
2. Critério AG? – Nesta fase, o nó escolhido na fase 1 é avaliado para que se torne efetivamente o nó corrente. Esta avaliação consiste em verificar se o valor de F deste nó candidato é menor que o valor do nó corrente. Caso não seja, o módulo AG será acionado e o algoritmo passa para a fase 5 (critério para utilização do módulo AG).
3. Escolha - Caso o nó candidato tenha o valor de F menor, este será o novo nó corrente.

4. Critério de parada? – Verifica se o nó corrente corresponde ao nó destino da busca. Caso seja, o algoritmo retorna o caminho encontrado, senão continua iterando.
5. Módulo AG – Este módulo é responsável por retornar um caminho compostos por nós. Estes são adicionados à lista fechada e o último nó do caminho passa a ser o novo nó corrente. Caso este não retorne nenhuma solução, retorna-se à fase 3.

   O Fluxograma 2 mostra detalhadamente o módulo AG.



Fluxograma 2 - Módulo AG

1. Algoritmo Genético - Nesta fase o AG itera sobre uma população de soluções parciais. Os operadores genéticos (cruzamento e mutação) são aplicados à população inicial. São geradas 12 soluções (inicialmente quatro indivíduos, a partir destes mais quatro por cruzamento e quatro por mutação). Para todas as soluções, calcula-se o valor de *fitness*.

2. Existem soluções? - Nesta fase, após ordenar a população de acordo com o valor de FA, verifica se existe alguma solução viável na população. Caso exista, a primeira solução (a mais promissora) é escolhida para ir para a próxima fase. Caso não exista nenhuma solução viável na população, o módulo encerra sua execução e não retorna nada para o módulo principal (BFS).
3. Validação - A solução escolhida previamente passa por esse processo que determina se esta é uma solução válida, inválida ou se deve passar pelo processo de adaptação.
4. É uma boa solução? – Após passar pela fase 4, a solução é rotulada e, de acordo com esse rótulo (válida, inválida ou a ser adaptada), será ou não retornada para o módulo principal (BFS). Caso seja inválida ou adaptada o algoritmo retorna à fase 2. Se válida passa para a próxima fase.
5. Aplicar solução – Nesta fase a solução parcial mais apta é transformada em um caminho composto nós. O módulo é encerrado e o caminho retornado para o BFS.

Para maiores detalhes, o pseudocódigo do PPGA encontra-se no Apêndice A.

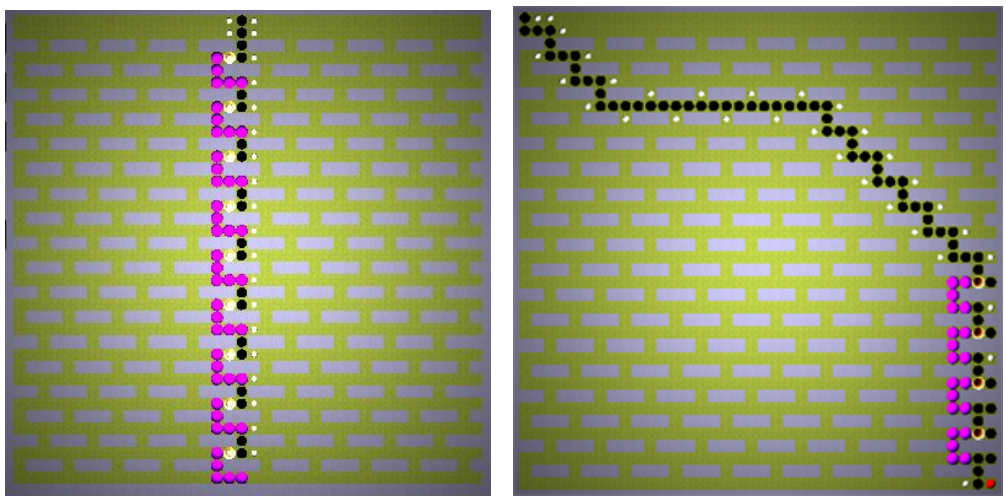
4. EXPERIMENTOS

Os experimentos têm como objetivo validar a proposta do trabalho. Para tanto, esta foi submetida a testes em determinados ambientes. Para as implementações dos algoritmos bem como a dos ambientes de testes foi escolhido uma linguagem de programação e uma plataforma de desenvolvimento. Estes experimentos deram origem a duas publicações.

4.1 TECNOLOGIA

Foi utilizada a plataforma Unity 3D, que permite uma representação visual dos experimentos. Unity 3D, é uma plataforma de desenvolvimento muito utilizada na indústria de jogos. Esta conta com vários componentes que facilitam o desenvolvimento de jogos eletrônicos, desde componentes que simulam a física, até componentes gráficos.

A plataforma Unity 3D possibilita a utilização da linguagem de programação C#, a qual foi utilizada para a implementação dos algoritmos.



(a) (b)

Figura 8 - Exemplo de execuções do PPGA na plataforma Unity 3D
Fonte: Santos et al (2012b)

A Figura 8 mostra dois exemplos de execuções feitas do algoritmo proposto, no ambiente desenvolvido na plataforma Unity 3D. As bolinhas da cor preta e branca representam os nós acessados pelo BFS, enquanto as bolinhas roxas, os nós que compões os caminhos gerados pelas soluções parciais do módulo AG.

A Figura 8(a) apresenta um padrão identificado pelo PPGA. As soluções parciais utilizadas são todas iguais, comprovando a identificação do padrão. Já a Figura 8(b), o módulo AG foi utilizado apenas em parte da busca, isso se deve ao fato de que o critério para a utilização do mesmo não foi atingido, durante parte da busca.

4.2 TESTES

Os testes tiveram o objetivo de gerar valores quantitativos para comparar o algoritmo proposto, PPGA, com dois métodos clássicos, A* e BFS. Os valores são referentes à quantidade de nós armazenados nas listas abertas e fechadas dos algoritmos comparados, a fim de mostrar o desempenho do PPGA, com o foco no consumo de memória exigido pelo armazenamento dos nós nas listas.

Os algoritmos foram executados em ambientes com as mesmas dimensões, 80X80 nós, e com 5 destinos cada. Ao chegar a cada destino, as listas abertas e fechadas dos algoritmos são zeradas e, os mesmos, buscam o próximo destino. Os valores referentes às quantidades de nós armazenados nas listas foram somados para cada destino.

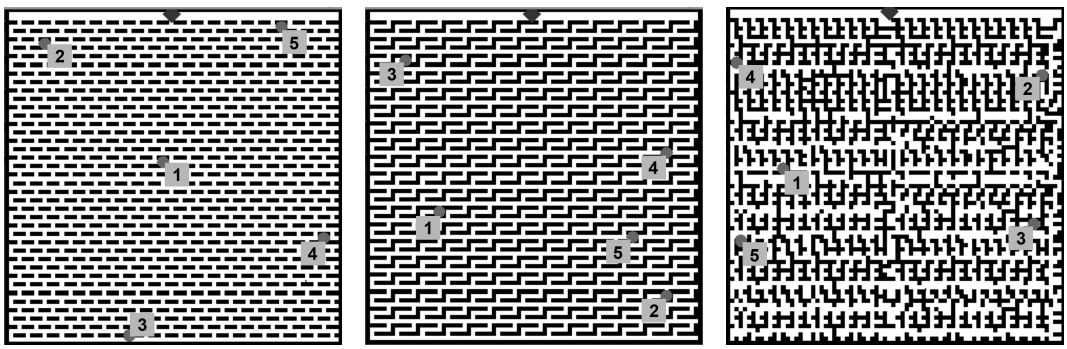
O PPGA mantém, durante as buscas de cada destino, uma memória. Esta é a solução parcial utilizada nas buscas dos destinos anteriores. A cada busca realizada pelo algoritmo no ambiente, este gera um aprendizado que é utilizado para próximas buscas no decorrer da execução.

Para cada ambiente, os algoritmos A* e BFS foram executados uma única vez, e os valores obtidos, guardados para a análise. Em contrapartida, por não se tratar de um algoritmo determinístico, o PPGA foi executado 20

vezes e uma média dos valores obtidos para cada execução foi calculada a fim de ser utilizada na análise.

Foram determinados três ambientes, com as disposições dos obstáculos distintas. São eles:

- Ambiente com um padrão de obstáculos, Figura 9(a);
- Ambiente misto, com mais de um padrão de obstáculos, Figura 9(b);
- Ambiente sem padrão, este não apresenta nenhum tipo de padrão de obstáculos, Figura 9(c).



(a) (b) (c)

Figura 9 - Ambientes de teste
Fonte: Santos et al (2012b)

Cada ambiente foi determinado para observar o comportamento do algoritmo proposto, nos seguintes casos:

- Onde o ambiente seja favorável (melhor caso) para o PPGA, um ambiente com um padrão, Figura 9(a);
- Onde o ambiente seja equilibrado, não favorecendo nenhum algoritmo testado (caso médio), Figura 9(b);
- Onde o ambiente seja desfavorável (pior caso) para o PPGA, Figura 9(c).

4.3 RESULTADOS E ANÁLISE

A proposta deste trabalho é voltada para o consumo eficiente de memória em problemas de busca de caminhos, na qual foi desenvolvida uma modificação para o algoritmo clássico BFS. A principal métrica analisada nos testes realizados foi a quantidade de nós armazenados nas listas de controle (abertas e fechadas) dos algoritmos comparados.

O algoritmo PPGA utiliza o AG, o qual consegue gerar soluções parciais que se adaptam ao ambiente. No entanto, este só consegue ter vantagens em ambientes cujos obstáculos estejam dispostos e tenham formatos padronizados. O resultado dos testes comprovaram esta afirmação.

A Tabela 7 mostra os valores obtidos nos testes, observa–se que o algoritmo A*, apresenta valores muito altos. Isso se deve ao fato deste ser um método que visa encontrar a solução ótima, enquanto o PPGA e o BFS visam apenas encontrar uma solução.

Tabela 7 - Resultados dos testes

	Ambiente		
Algoritmo	**Com Padrão**	**Misto**	**Sem Padrão**
A*	16016	27692	18468
BFS	2048	3088	5796
PPGA	1725	3142	6252

Fonte: Santos et al (2012b)

O Gráfico 1 apresenta uma comparação apenas entre o BFS e o PPGA, devido a ambos não visarem encontrar o melhor caminho.

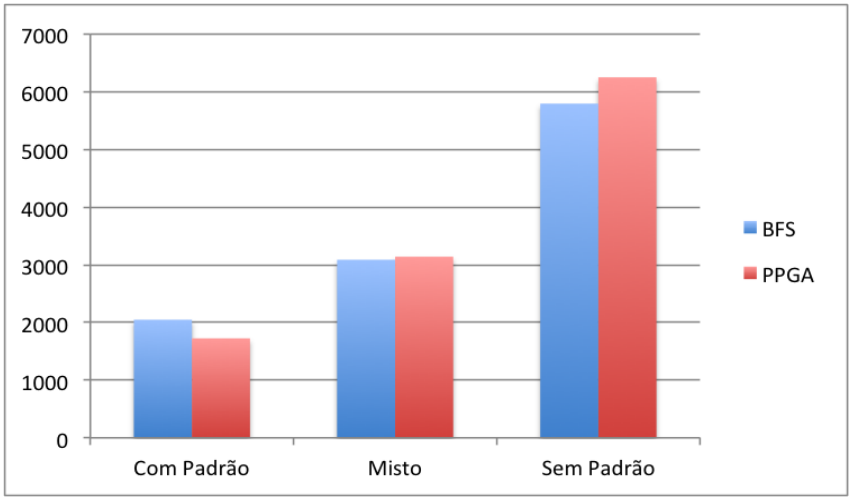


Gráfico 1 - Gráfico comparativo
Fonte: Santos et al (2012b)

Analisando o gráfico, pode-se observar que, em ambientes sem padrão, o PPGA, teve uma perda considerável, cerca de 8%, em relação ao BFS. Esta perda se deve ao fato do AG não conseguir reaproveitar as soluções parciais utilizadas anteriormente e, devido a este fato, há um desperdício de processamento e armazenamento.

No caso médio (ambiente misto), o PPGA mostra-se equilibrado em relação ao algoritmo comparado.

Já no ambiente padronizado, o PPGA, obteve uma redução de 16% na quantidade de nós armazenados, em relação ao BFS. Este resultado comprova a vantagem que o algoritmo proposto tem neste tipo de ambiente.

5. CONCLUSÃO E TRABALHOS FUTUROS

Este trabalho propôs um método que visa a identificação do padrão dos obstáculos, com o intuito de otimizar a busca de caminhos, como em Demyenand e Buro (2006). A modelagem das soluções tem certa semelhança com a modelagem utilizada por Burchardt e Salomon (2006), diferenciando desta por permitir uma flexibilidade e adaptação ao ambiente. Diferentemente de Leigh et al. (2007), que propuseram um método que utiliza o AG com um treinamento *off–line*, a proposta aqui apresentada utilizou um treinamento *on-line*, tornando–a mais utilizável.

No trabalho de Machado et al. (2011) é mostrado que a redução dos nós acessados e armazenados é proporcional ao ganho no processamento. Com isso o algoritmo por eles proposto, RTP-GA, é focado no processamento em tempo real. Após algumas análises este método mostrou–se inviável para requisitos de tempo real. A abordagem apresentada com o PPGA aproveita a redução dos nós armazenados proporcionada no RTP-GA, melhorando–a e mudando o foco para apenas o consumo eficiente de memória.

O algoritmo proposto, PPGA, se baseou em uma mescla de duas técnicas, o BFS, que é um método heurístico de busca e o AG, que é uma metaheurística. O primeiro foi utilizado para conduzir a busca enquanto o segundo foi responsável por gerar soluções para superar as adversidades do ambiente (obstáculos). O AG, juntamente com a modelagem das soluções, mostrou uma boa adaptabilidade em ambientes com padrão de obstáculos.

Os resultados obtidos nos testes, mostraram que, devido a sua principal funcionalidade (identificação de padrões), o PPGA se mostrou mais eficiente em ambientes cuja disposição e tamanho dos obstáculos sejam padronizados. Nesses ambientes, em comparação com o BFS clássico, obteve uma redução de 16% dos nós armazenados nas listas aberta e fechada.

Os testes feitos nos ambientes misto e sem padrão, mostraram que o método proposto tem muito que ser melhorado. Essas melhorias e otimizações serão propostas para trabalhos futuros. Outro assunto que deverá ser abordado em trabalhos futuros será a comparação com outros métodos de busca que visam também o consumo eficiente de memória.

PUBLICAÇÕES

SANTOS, U. O.; MACHADO, A. F. V. ; CLUA, E. W.; ***Pathfinding Based on Pattern Detection Using Genetic Algorithms***. In: Simpósio Brasileiro de Jogos e Entretenimento Digital (SBGames), Brasília, DF. 2012a.

SANTOS, U. O.; MACHADO, A. F. V. ; CLUA, E. W.; ***Best First Search with Genetic Algorithm for Space Optimization in Pathfinding Problems***. In: GAMEON Conference, Málaga, Espanha. 2012b.

REFERÊNCIAS

BJÖRNSSON, Yngvi, ENZENBERGER, Markus, HOLTE, Robert, SCHAEFFER, Jonathan. ***Fringe search: Beating A* at Pathfinding on Computer Game Maps***. In: *Proceedings of the IEEE Symposium on Computational Intelligence in Games (CIG'05), April 4-6, Colchester, Essex, UK*, pp. 125–132, 2005.

MATTHEWS, James, ***Basic A* Pathfinding Made Simple*****,** *AI Game Programming Wirdom,* Charles River Media, 2002.

CAIN, Timothy, ***Practical Optimizations for A* Path GERAÇÃO,*** *AI Game Programming Wirdom,* Charles River Media, 2002.

HIGGINS, Daniel F., ***Pathfinding Design Architecture,*** AI Game Programming Wisdom, Charles River Media, 2002.

HOLLAND, J.H. ***Outline for a logical theory of adaptive systems***. J. Assoc. Comput. Mach., vol. 3, pp. 297-314, 1962.

DECHTER, Rina; PEARL, Judea; ***Generalized Best-First Search Strategies and the Optimality of A*.*** Journal of the Association for Computing Machinery, Vol. 32, No. 3, July 1985, pp. 505-536.

LEIGH, Ryan; LOUIS, Sushil J.; MILES, Chris; ***Using a Genetic Algorithm to Explore A*-like Pathfinding Algorithms*****.** In: IEEE Congress on Computational Intelligence and Games. CIG – 2007.

DEMYEN, Douglas; BURO, Michel; ***Efficient Triangulation-Based Pathfinding***. In: AAAI'06 Proceedings of the 21st national Conference on Artificial intelligence. 2006.

BURCHARDT, H.; SALOMON, R. ***Implementation of Path Planning using Genetic Algorithms on Mobile Robots***. IEEE Congress on Evolutionary Computation. CEC 2006.

HART, P.E.; NILLSON, N.J.; RAPHAEL, B.; ***A formal basis for the heuristic determination of minimum cost paths***. IEEE Transactions on Systems Science and Cybernetics, 1968.

BJORNSSON, Y.; ENZENBERGER, M.; HOLTE, R.C.; SCHAEFFER, J.; ***Fringe search: Beating A* at pathfinding on gamemaps***. IEEE Computational Intelligence in Games, 2005.

STODOLA, P.; MAZAL, J.; ***Optimal location and motion of autonomous unmanned ground vehicles***. In World Scientific and Engineering Academy and Society. WSEAS 2010.

MITCHELL, M.; ***An introduction to genetic algorithms***. The MIT Press, United States; 1996.

SOARES, G. L., **Algoritmo Genético: Estudo, Novas Técnicas e Aplicações**, Dissertação de Mestrado, Programa de Pós-Graduação em Engenharia Elétrica, Univer- sidade Federal de Minas Gerais, Belo Horizonte, Brasil, 1997.

TANOMARU, J. (1995). **Motivação, fundamentos e aplicações de algoritmos genético**, *II Congresso Brasileiro de Redes Neurais - III Escola de Redes Neurais*.

KORF, R. ***Depth-first iterative deepening: An optimal admissible tree search***. Artificial Intelligence, 1985.

RUSSELL, Stuart. J.; NORVIG, Peter.; ***Artificial Intelligence: A Modern Approach***, Second Edition, 2004.

MACHADO, A. F. V. ; CLUA, E. W. ; GONÇALVES, R. ; SANTOS, U. O. ; NEVES, T.; OCHI, L. S.; ***Real Time Pathfinding with Genetic Algorithm***. In: Simpósio Brasileiro de Jogos e Entretenimento Digital (SBGames), Salvador, BA. 2011.

UNITY TECHNOLOGIES: ***Unity 3D User Manual***. Disponível em: <www.unity3d.com/support/documentation/>.

APÊNDICE A - PSEUDOCÓDIGO

.

PPGA()

```
01. LISTA_ABERTA ← VAZIO
02. LISTA_FECHADA ← VAZIO
03. NO_CORRENTE ← INICIO
04. SUCESSO ← 0
05. NOT_SUCESSO ← 0
06. SUCESSO_TAXA ← 0
07. DISTANCIA_TAXA ← 0
08.   MELHORES_INDIVIDUOS ← VAZIO
09. enquanto NO_CORRENTE não for o DESTINO faça
10.           adicionar(LISTA_FECHADA,NO_CORRENTE)
11.           para cada vizinho do NO_CORRENTE faça
12.                   se vizinho não está na LISTA_FECHADA
                              e não está na LISTA_ABERTA então
13.                           DISTANCIA ← manhattan(vizinho,DESTINO)
14.                           gerarCusto(vizinho,DISTANCIA)
15.                           gerarPai(vizinho,NO_CORRENTE)
16.                           adicionar(LISTA_ABERTA,vizinho)

17.                   fim_se
18.           fim_para
19.           NO ← remove item de menor custo da LISTA_ABERTA
20.           SUCESSO ← SUCESSO/ NAO_SUCESSO
21.           DISTANCIA_TAXA ←
                      manhattan(INICIO,DESTINO)/manhattan(NO_CORRENTE,DESTINO)
22.           se AmbienteComPadrão(SUCESSO_TAXA,DISTANCIA_TAXA) igual a 1   então
23.                   se o custo do NO for maior que o custo  do NO_CORRENTE então
24.                           MELHORES_INDIVIDUOS ←
                                    MODULO_AG(MELHORES_INDIVIDUOS, NO_CORRENTE,
                                    DESTINO)
```

```
25.                         se MELHORES_INDIVIDUOS  não VAZIO então
26.                                 LISTA_NOS ←
27.                                 converte MELHORES_INDIVIDUOS para lista de nós
28.                                 adiciona nos da LISTA_NOS na LISTA_FECHADA
29.                                 NO ← pegue o ultimo item da LISTA_NOS
30.                                 SUCESSO++
31.                         senão
32.                                 NÃO_SUCESSO ++
33.                         fim_se
34.                 fim_se
35.
36.         fim_se
37.         NO_CORRENTE ← NO
38.
39. fim_enquanto
```

MODULO_AG(MELHORES_INDIVIDUOS, NO_CORRENTE, DESTINO)

```
01. GERAÇÃO ← VAZIO
02.   gerarPopulaçãoInicial(GERAÇÃO, MELHORES_INDIVIDUOS,NO_CORRENTE)
03.   cruzamento(GERAÇÃO,NO_CORRENTE)
04.   mutação(GERAÇÃO,NO_CORRENTE)
05.   calcularFAeOrdenar(GERAÇÃO,DESTINO)
06.   para cada individuo da GERAÇÃO faça
07.         se validaçãoNoAmbiente(individuo)  então
08.                 retorna individuo
09.         senão
10.                 novo_individuo = adaptação(individuo)
11.                 adicionar novo_individuo na GERAÇÃO
12.         fim_se
13.   fim_para
14.   retorna VAZIO
```