UNIVERSIDADE FEDERAL DE SANTA MARIA
CENTRO DE TECNOLOGIA
GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

Caroline Chagas

ALGORITMOS DE BUSCA DE CAMINHOS VOLTADOS PARA INFORMAÇÕES DE ALTURA E INCLINAÇÃO REPRESENTADAS EM MAPAS DE NAVEGAÇÃO

Caroline Chagas

ALGORITMOS DE BUSCA DE CAMINHOS VOLTADOS PARA INFORMAÇÕES DE ALTURA E INCLINAÇÃO REPRESENTADAS EM MAPAS DE NAVEGAÇÃO

Trabalho de Conclusão de Curso apresentado ao Programa de Graduação em Ciência da Computação da Universidade Federal de Santa Maria (UFSM), como requisito parcial para a obtenção do título de **Bacharel em Ciência da Computação**.

Orientador: Luís Alvaro de Lima Silva

457

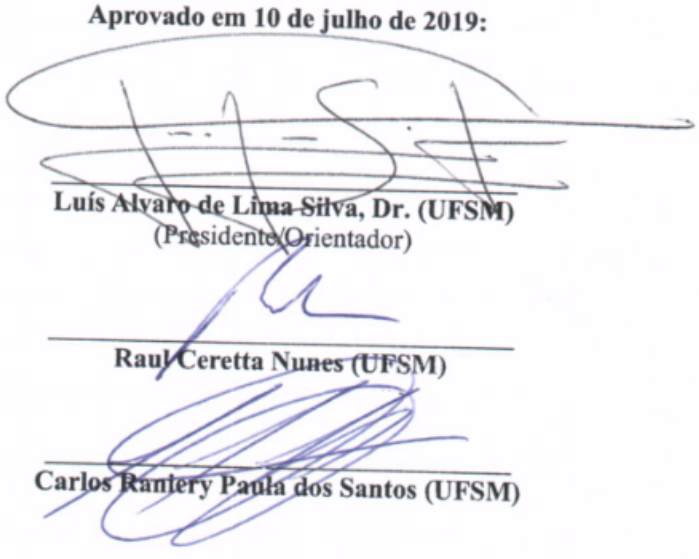
Santa Maria, RS
2019

Caroline Chagas

ALGORITMOS DE BUSCA DE CAMINHOS VOLTADOS PARA INFORMAÇÕES DE ALTURA E INCLINAÇÃO REPRESENTADAS EM MAPAS DE NAVEGAÇÃO

Trabalho de Conclusão de Curso apresentado ao Programa de Graduação em Ciência da Computação da Universidade Federal de Santa Maria (UFSM), como requisito parcial para a obtenção do título de **Bacharel em Ciência da Computação**.

Aprovado em 10 de julho de 2019:



Luís Alvaro de Lima Silva, Dr. (UFSM)
(Presidente/Orientador)

Raul Ceretta Nunes (UFSM)

Carlos Rantery Paula dos Santos (UFSM)

Santa Maria, RS
2019

AGRADECIMENTOS

Este trabalho foi concretizado com empenho e dedicação, tendo como objetivo sua realização com êxito. Esta é uma grande conquista, que me aproxima ainda mais de um sonho: ser graduada em Ciência da Computação. Esta conquista foi possível com a participação de pessoas fundamentais em minha jornada na área da Computação.

Primeiramente, gostaria de agradecer à minha família, principalmente aos meus pais, por todo apoio, educação, carinho e incentivo aos meus estudos e especializações desde cedo. Isso inclui o incentivo a ter dedicação desde a escola quanto à participação em cursos de idiomas e cursos profissionalizantes, além do curso técnico integrado em informática, que me inseriu na área da Computação. Sempre me auxiliaram de todas as formas, fosse financeiramente ou psicologicamente, me fortalecendo, motivando e amparando, proporcionando o que fosse necessário para minha formação e também bem-estar. Agradeço também à minha avó, Erdila (*in memorian*), pelo apoio e preocupação com meus estudos e minha estadia em Santa Maria, buscando me ajudar sempre que possível. A participação de minha família foi essencial, a minha conquista é a nossa conquista!

Gostaria de agradecer aos meus professores da graduação, por todo conhecimento compartilhado e auxílio no aprendizado, além da disponibilidade em auxiliar no que fosse necessário. Em especial, agradeço ao meu orientador Luís Alvaro de Lima Silva, pela orientação neste trabalho, atenção e ensinamentos. Ainda, o agradeço por ter me selecionado para desenvolver pesquisa em sua equipe. Foi uma experiência que contribui muito para minha formação. Agradeço aos professores Raul Ceretta Nunes e Carlos Raniery, pela disponibilidade de participarem da minha banca avaliadora, contribuindo para o aperfeiçoamento deste trabalho.

Agradeço também, aos professores do IFSul Câmpus Venâncio Aires e às oportunidades que o instituto federal me proporcionou. Foi a partir do ingresso no técnico em informática que meu interesse na área aumentou e resultou na decisão de minha carreira profissional. Além disso, a experiência de participar de eventos internacionais desde cedo em minha vida acadêmica, desenvolver pesquisa e extensão a participar de feiras e mostras tanto científicas quanto culturais, foram únicas e fundamentais em minha formação acadêmica e pessoal.

Ainda, agradeço à Cantu&Stange, empresa na qual realizei meu primeiro estágio, pela oportunidade profissional tanto de realização de estágio na empresa quanto pelo convite para fazer parte da equipe como desenvolvedora de software, sendo então, contratada. Agradeço

em especial ao nosso supervisor, Carlos Stange, pela atenção, oportunidade e conhecimentos transmitidos. O estágio na empresa foi decisivo para que eu seguisse na área.

Não poderia deixar de agradecer também aos meus amigos do curso, que me acompanham desde 2015, ano no qual ingressei na UFSM. Agradeço a companhia, parceria e prestatividade de todos, em especial aos meus amigos Daniel, Gabriel e Vinícius, que foram diferenciais em minha jornada na universidade e em Santa Maria. São aqueles nos quais sei que posso confiar e me ajudaram muito durante a graduação, tanto enquanto colegas como amigos com os quais pude contar.

Agradeço também à equipe de Inteligência Artificial do projeto SIS-ASTROS, que me auxiliaram na adaptação ao projeto, no aprendizado e também na realização de minhas pesquisas e deste trabalho. Em especial, agradeço aos colegas Eliakim e Daniel, com quem convivo diariamente no projeto e me auxiliaram muito durante todo o desenvolvimento deste trabalho.

RESUMO

ALGORITMOS DE BUSCA DE CAMINHOS VOLTADOS PARA INFORMAÇÕES DE ALTURA E INCLINAÇÃO REPRESENTADAS EM MAPAS DE NAVEGAÇÃO

AUTOR: Caroline Chagas
ORIENTADOR: Luís Alvaro de Lima Silva

A Inteligência Artificial é uma área de pesquisa em evidência atualmente, a qual abrange diversos ramos de estudo. Dentre suas ramificações, compreende-se a área que visa solucionar problemas de busca de caminhos, conhecidos como problemas de *pathfinding*. O trabalho aborda o problema de busca de caminhos em mapas de navegação, priorizando a computação de caminhos quando informações de altura e inclinação no terreno são consideradas. Quando se trata de ambientes montanhosos, o desempenho dos agentes pode ser comprometido à medida que os desníveis interfiram em sua movimentação. Em sistemas de simulação, contexto de sistema ao qual o algoritmo do trabalho é direcionado, busca-se simular eventos da forma mais próxima possível da realidade, portanto uma falha poderia comprometer a execução do sistema. Assim, o tratamento de possíveis riscos durante a movimentação, fornecidos pelos desníveis do terreno, é primordial para o bom funcionamento e integridade do sistema. Concluída a implementação do algoritmo proposto, os experimentos para testar sua eficiência foram conduzidos em dois terrenos virtuais (que remetem a terrenos reais), com diferentes características de relevo. Ainda, o algoritmo foi comparado a outros três algoritmos com características semelhantes para realizar sua avaliação. Desta forma, este trabalho fornece como contribuições o desenvolvimento de um algoritmo que planeja rotas utilizando uma técnica denominada “campo de visão” ao passo em que trata altura, uma abordagem não encontrada na literatura. Assim, o algoritmo proporciona caminhos suavizados que evitam regiões montanhosas, cujas inclinações possam fornecer riscos à movimentação dos agentes envolvidos na simulação. Ainda, o algoritmo utiliza técnica de busca hierárquica, utilizando pré-processamento de caminhos entre nodos importantes na composição da hierarquia da estrutura de representação do terreno virtual, o que resulta em uma otimização do tempo de execução.

Palavras-chave: Busca de caminho. Altura. Inclinação. Busca hierárquica de caminho. Inteligência artificial. Navegação. Sistema de simulação.

ABSTRACT

PATHFINDING ALGORITHMS FOR HEIGHT AND SLOPE INFORMATION TREATMENT, REPRESENTED ON NAVIGATION MAPS

AUTHOR: Caroline Chagas
ADVISOR: Luís Alvaro de Lima Silva

Artificial Intelligence is an area of research in evidence nowadays, which covers several fields of study. Among its ramifications, it is understood the area that seeks to solve path finding problems. This work addresses the problem of searching for paths in navigation maps, prioritizing the computation of paths when height and slope information in the terrain are considered. When it comes to mountainous environments, agent performance can be compromised as uneven in their movement. In simulation systems, the system context to which the work algorithm is directed, it is sought to simulate events as closely as possible to reality, so a failure could compromise the execution of the system. Thus, the treatment of possible risks during the movement, provided by the unevenness of the terrain, is paramount for the good functioning and integrity of the system. After the implementation of the proposed algorithm, the experiments to test its efficiency were conducted in two virtual terrains (with real terrains), with different relief characteristics. Furthermore, the algorithm was compared to three other algorithms with similar characteristics to perform its evaluation. As input, this paper provides as contributions the development of an algorithm that plan routes using a technique called "line of sight" while it treats height, an approach not found in the literature. Thus, the algorithm provides smoothed paths that avoid mountainous regions, whose inclinations may provide risks to the movement of the agents involved in the simulation. Also, the algorithm uses hierarchical search technique, using preprocessing of paths between important nodes in the hierarchy composition of the virtual terrain representation structure, which results in an optimization of execution time.

Keywords: Pathfinding. Height. Slope. Hierarquical pathfinding. Artificial Intelligence. Navigation. Simulation Systems.

LISTA DE FIGURAS

LISTA DE ALGORITMOS

LISTA DE TABELAS

SUMÁRIO

1 INTRODUÇÃO

A Inteligência Artificial é uma área de pesquisa, a qual abrange diversos ramos de estudo. Dentre suas ramificações, compreende-se a área que visa solucionar problemas de busca de caminhos, conhecidos como problemas de *pathfinding*. Em geral, um problema de busca de caminhos é caracterizado pela busca pelo melhor caminho de acordo com um determinado critério (menor caminho, menor custo, entre outros), considerando uma estrutura de dados que representa os caminhos e obstáculos do ambiente virtual (mapa de navegação). Assim, busca-se explorar o algoritmo mais adequado para solucionar o problema proposto. Ainda, os problemas abordados por esses algoritmos envolvem os denominados agentes, que são os indivíduos envolvidos no problema. Estes agentes realizam raciocínio via algoritmos simbólicos, procurando atingir metas ao passo em que percebem seu ambiente (Russell e Norvig, 2016).

Em sistemas de simulação, contexto de sistema ao qual o algoritmo do trabalho é direcionado, busca-se simular eventos da forma mais próxima possível da realidade. Não utilizar os algoritmos adequados aos problemas aplicados ao sistema e adequados também ao comportamento desejado para os agentes pode implicar em falhas, as quais podem comprometer a execução do sistema. Em geral, os algoritmos que traçam rotas para agentes funcionam adequadamente para superfícies planas. No entanto, quando se trata de ambientes montanhosos, o desempenho dos agentes pode ser comprometido à medida que os desníveis interfiram em sua movimentação, de forma que determinadas inclinações possam promover deslizamentos, tombamentos de veículos ou simplesmente impossibilitar o movimento. Assim, o tratamento destes possíveis riscos no planejamento das rotas dos agentes envolvidos é primordial para o bom funcionamento e integridade do sistema, priorizando a computação de caminhos quando informações de altura e inclinação no terreno são consideradas. Em geral, a forma mais comum de solucionar um problema que envolva desníveis de relevo é bloquear nodos que indiquem dificuldades ou obstrução do caminho. Todavia, este tipo de tratamento pode não ser o mais adequado de acordo com o contexto do problema a ser tratado.

No contexto do desenvolvimento de sistemas de simulação, o trabalho considera a representação do terreno em uma *quadtree* hierárquica (Samet, 1984), o que ainda é uma estrutura pouco explorada por algoritmos de busca de caminhos em ambientes virtuais simulados.

Os requisitos e aspectos do problema proposto bem como a necessidade de implementação de um algoritmo eficiente no tratamento de altitude e suavização de caminhos encontrados, além de considerar a estrutura de representação do terreno sendo irregular e hierárquica, tornam o trabalho a ser desenvolvido relevante e diferenciado.

Além disso, este trabalho fornece como contribuições o desenvolvimento de um algoritmo que planeja rotas utilizando uma técnica denominada "campo de visão", a ser detalhadamente explanada na seção 2, ao passo em que trata altura. Desta forma, as contribuições envolvem a associação dos benefícios de diversas técnicas em um único algoritmo:

- Suavização de caminho, que proporciona maior qualidade, ou seja, caminhos mais curtos;
- Utiliza o campo de visão modificado, com o custo elaborado;
- Balanço entre caminho mais curto e menos custoso (menor dificuldade de travessia);
- Utiliza busca hierárquica com pré-processamento, otimizando o tempo de execução.

O uso de campo de visão tratando informações que remetam a alguma restrição de altura, sem utilizar apenas o bloqueio ou não de nodos, durante a busca de caminho (muitas vezes convertidas em pesos durante este processo), é uma abordagem não encontrada na literatura. Assim, o algoritmo proporciona caminhos suavizados que evitem regiões montanhosas, cujas inclinações possam fornecer riscos à movimentação dos agentes envolvidos na simulação. Ainda, o algoritmo utiliza técnica de busca hierárquica, utilizando pré-processamento de caminhos entre nodos importantes na composição da hierarquia da estrutura de representação do terreno virtual, o que será abordado na Seção 3.

1.1 JUSTIFICATIVA

O projeto justifica-se pela necessidade prática de um algoritmo de busca pelo melhor caminho aplicado ao contexto de simulação tática militar, levando em consideração as particularidades de altura do terreno, priorizando o tratamento de inclinações ao longo do caminho. Isso envolve questões referentes às limitações de movimentação de agentes envolvidos no problema proposto, buscando evitar trajetos que possam oferecer riscos aos mesmos, como impedir movimento destes agentes devido à dificuldade causada pela

complexidade do terreno. Visto que a simulação busca reproduzir atividades que ocorrem no mundo real, executá-las de forma a assemelhar-se o mais próximo possível ao realismo, torna-se um importante objetivo. Portanto, o algoritmo deve fornecer caminhos suavizados e que, ainda, proporcionem segurança durante a navegação, evitando falhas na execução da simulação.

1.2 OBJETIVOS

1.2.1 Geral

O objetivo do trabalho é propor e testar um algoritmo de busca de caminho (*pathfinding*) executado em mapas virtuais de navegação utilizados em sistemas de simulação. A implementação deste algoritmo trata questões referentes à altura do terreno (relevo), buscando o menor e melhor caminho, de acordo com o critério de evitar inclinações que prejudiquem a movimentação do agente, entre pontos de origem e destino determinados. Além disso, pretende-se buscar caminhos otimizados, onde pequenos desvios nas rotas sendo analisadas sejam suavizados, e que o caminho a ser escolhido trafegue por rotas que permitam o movimento seguro de agentes em regiões de montanhas. Os algoritmos desenvolvidos são parametrizados de modo que limitações de movimentação de diferentes tipos de agentes sejam consideradas, como limitações de movimentação de veículos, leves e pesados, e de pessoas envolvidas nas simulações realizadas. Além disso, os algoritmos de busca de caminho consideram a otimização do tempo de execução, assim não sobrecarregando as demais simulações sendo desenvolvidas no sistema de simulação.

1.2.2 Específicos

- Apresentar uma revisão da literatura à respeito de algoritmos de busca de caminhos;
- Descrever formas de buscar caminhos que considerem a representação de altura em mapas de navegação de terrenos virtuais usados em sistemas de simulação;
- Pesquisar, desenvolver e testar um algoritmo de busca de caminhos voltado para o tratamento de altura e inclinações em terrenos virtuais simulados;
- Abordar o problema de busca de caminhos em terrenos montanhosos, de forma que atenda às necessidades para movimentação adequada e segura de diferentes tipos de agentes;

- Tornar caminhos encontrados mais realistas aplicando algoritmos de suavização em caminhos sendo investigados.

2 REVISÃO BIBLIOGRÁFICA

Visando adquirir embasamento para a realização do trabalho sobre os assuntos abordados neste - como estruturas de representação de mapas de navegação e suas topologias, representação de informações sobre o terreno e algoritmos de busca de caminho -, foi realizada uma revisão bibliográfica na literatura. Uma apresentação sobre problemas de busca de caminho é feita na seção 2.1. Na seção 2.2, são abordadas as topologias de representação de terrenos virtuais, que podem ser baseadas tanto em estruturas irregulares quanto regulares. Técnicas que auxiliam no processo de busca são apresentadas da seção 2.3 à seção 2.6.

2.1 O PROBLEMA DA BUSCA DE CAMINHO

Em sistemas de simulação, bem como em jogos digitais, um dos requisitos mais básicos para agentes ou personagens, que incumbem à Inteligência Artificial, é permitir que possam navegar com êxito pelo terreno virtual. Para isso, são desenvolvidos e implementados os algoritmos de busca de caminho. Estes algoritmos buscam rotas trafegáveis para agentes baseados em algum critério de prioridade, tais como priorizar caminho mais curto, caminho de menor custo, entre outros, dados os pontos de origem e destino determinados no terreno.

Dentre os algoritmos de busca, o algoritmo A* (Nilsson e Nilsson, 1998) e suas variações estão entre os mais explorados. A* é um algoritmo baseado em heurística, o qual é otimamente eficiente tal como discutido em (Dooms, 2013), garantindo menor quantidade de nodos expandidos durante a navegação em relação a outros algoritmos. Apesar de ser um algoritmo ótimo e completo, muito explorado em diversos âmbitos da computação, principalmente na área de jogos, seu custo computacional pode ficar elevado à medida que a complexidade do problema aumenta, como, por exemplo, aumentando o espaço de busca, a quantidade de agentes, entre outras formas.

O algoritmo A* utiliza uma heurística para auxiliar a ordem em que os nodos que representam a estrutura do terreno são processados, com o intuito de diminuir o tempo de processamento. Sua execução ocorre baseada na seguinte lógica: o nodo inicial do caminho é adicionado a uma lista, a qual mantém os nodos abertos pelo algoritmo ao longo de sua execução. Enquanto houver nodos nesta lista, o algoritmo escolhe o melhor nodo e verifica se este é um nodo objetivo. Se este for o destino da busca, o caminho foi encontrado. Caso não seja, o algoritmo busca por vizinhos do nodo atual ainda não visitados. Para cada um destes, é

verificado o custo de movimento do ponto inicial até a posição atual e o custo de movimento para o vizinho.

O algoritmo A* é baseado nas seguintes funções de custo, as quais definem a prioridade de processamento de um nodo n: $f(n) = g(n) + h(n)$. Estas são definidas da seguinte forma:

- **g(n):** é função de avaliação do custo para chegar até o nodo atual;
- **h'(n):** é a função de avaliação que estima o custo para se chegar deste nó até o estado final. Esta função é a responsável pela parte "heurística" do processamento. É chamada de h' para indicar que é uma aproximação da função de avaliação exata h do custo para chegar deste nó até o estado final.
- **f'(n):** é a soma de g e h'. É a avaliação global de um determinado nó.

Algoritmo 1: Algoritmo A*

```
1  Main()
2      g(s_start) := 0;
3      parent(s_start) := s_start;
4      open := ∅;
5      open.Insert(s_start, g(s_start) + h(s_start));
6      closed := ∅;
7      while open ≠ ∅ do
8          s := open.Pop();
9          if s = s_goal then
10             return "path found";
11         closed := closed ∪ {s};
12
13         [UpdateBounds(s)];
14         foreach s' ∈ nghbrs_vis(s) do
15             if s' ∉ closed then
16                 if s' ∉ open then
17                     g(s') := ∞;
18                     parent(s') := NULL;
19                 UpdateVertex(s, s');
20     return "no path found";
21 end
22 UpdateVertex(s,s')
23     if g(s) + c(s, s') < g(s') then
24         g(s') := g(s) + c(s, s');
25         parent(s') := s;
26         if s' ∈ open then
27             open.Remove(s');
28         open.Insert(s', g(s') + h(s'));
29 end
```

Fonte: (Nash *et al.*, 2007).

Para possibilitar a busca, uma estrutura de representação do terreno deve ser utilizada. Esta estrutura descreve as características do terreno/ambiente e o espaço de busca disponível. A revisão sobre topologias de terrenos é apresentada na próxima seção.

2.2 ESTRUTURAS DE REPRESENTAÇÃO DE MAPAS DE NAVEGAÇÃO

A representação do terreno é uma questão importante a ser considerada para realizar a busca de caminho, uma vez que o algoritmo de busca precisa ter a compreensão sobre a descrição do ambiente virtual no qual o *pathfinding* será realizado. Além disso, o ambiente de busca deve ser armazenado de forma a facilitar e simplificar o acesso aos dados armazenados na estrutura, durante o processo de busca de caminhos (Anguelov, 2011). A escolha do algoritmo ideal para resolver o problema de busca é realizada conforme os objetivos do desenvolvedor e da estrutura na qual o terreno virtual é representado.

2.2.1 Estruturas regulares

Estruturas regulares são mais fáceis de implementar, visto que sempre têm o mesmo números de nodos, ou células, e arestas, independentemente da quantidade de características do terreno a serem representadas no ambiente virtual (Souissi *et al.*, 2013). Esta topologia divide a estrutura em nodos/células na forma de polígonos regulares que possuam mesmo tamanho.

Apesar da praticidade de implementação e busca em estruturas regulares, elas podem não ser ideais para representar terrenos complexos. Por exemplo, regiões específicas de aglomerados de obstáculos podem necessitar maior refinamento dos nodos da estrutura de representação. Outra necessidade de representação diz respeito a considerar obstáculos com formatos diferentes em relação à forma do nodo, como obstáculos cuja forma seja circular ou triangular, enquanto o nodo é definido na forma quadrada. Neste caso, também é necessário maior refinamento da estrutura de representação. Uma vez que a estrutura seja regular, caso necessite refinamento em uma área específica do terreno devido à ocorrência de alguma característica ou obstáculo, todos seus nodos devem ser refinados, de forma que todos os nodos da estrutura possuam mesmas as dimensões. É fundamental notar que esse detalhamento na representação do terreno exige maior consumo de memória para realizar a execução dos algoritmos de busca de caminhos, além de maior tempo de execução a cada

busca. Assim, a busca de caminho em terrenos de grande escala utilizando esta estrutura pode apresentar tempos de execução e uso de memória insatisfatórios (Anguelov, 2011). Por outro lado, caso seja preferencial manter nodos ou células maiores, ou seja, não muito refinados, a precisão na representação do terreno pode ser prejudicada, não demonstrando correspondência com a realidade.

2.2.2 Estruturas irregulares

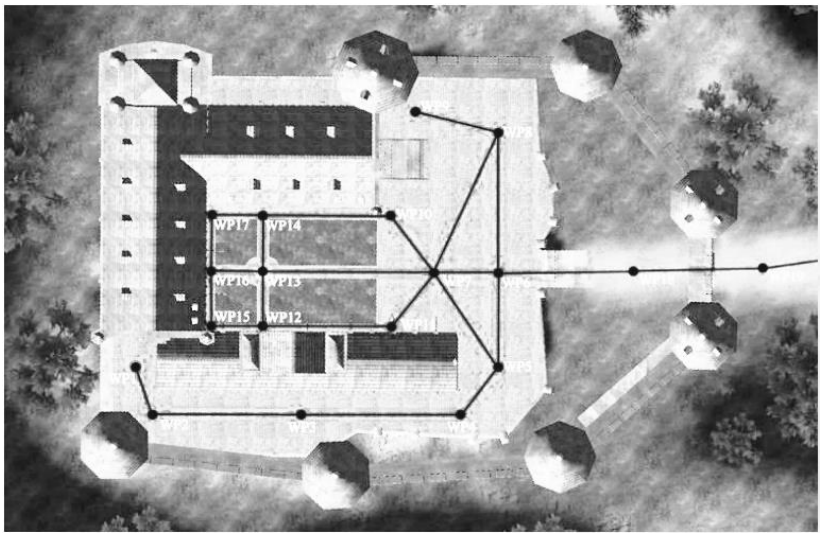
Estruturas irregulares são caracterizadas por terem nodos de diferentes tamanhos, além da possibilidade de apresentarem diferentes formas. Dessa forma, em um terreno com regiões que possuam diferentes características, é possível refinar apenas as áreas de interesse (pode-se refinar baseado em um determinado critério, como elevação do terreno, densidade de árvores, densidade de obstáculos específicos, entre outros parâmetros). Isso permite manter a caracterização e compatibilidade com o terreno original, sem sobrecarregar a memória usada para representar o terreno virtual ou prejudicar a busca por falta de precisão na estrutura de representação. Ao ser explorado pelo algoritmo de busca de caminho, essa estrutura geralmente exige menos uso de memória e possivelmente permite obter um menor tempo de execução em relação à execução de algoritmos de *pathfinding* em estruturas de representação regulares de terreno.

Como topologias consideradas irregulares, podemos citar grafos de waypoints, grafos de visibilidade e NavMesh (Kapadia e Badler, 2013). Como descrito por (Algfoor et al., 2015), representar o terreno utilizando uma *quadtree* também é uma forma de representação com estrutura irregular.

2.2.2.1 Waypoints

Nesta topologia, os chamados *waypoints* são dispostos pelo terreno virtual e em seguida são conectados para formar o grafo de navegação. São uma representação aproximada do terreno, sendo mais econômicas do que as malhas poligonais dos cenários, por isso tornam-se mais fáceis de armazenar e retornam rapidamente à busca de caminho. A construção dessa estrutura pode ser realizada tanto de forma manual quanto autônoma.

Figura 2.2.2.1.1: Representação de mapa virtual utilizando w*aypoints*.



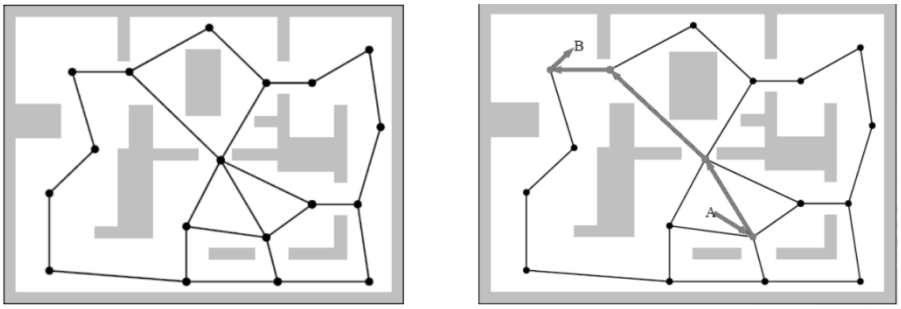
Fonte: Lima, E. S. **Waypoints e pathfinding – Inteligência artificial.** Disponível em: < edirlei.3dgb.com.br/aulas/ia_2012_1/IA_Aula_25_Waypoints_e_Pathfiding.pptx> Acesso em: 04 de jul de 2019.

2.2.2.2 Grafo de visibilidade

Criar um grafo de navegação baseado em pontos de visibilidade, formando o chamado grafo de visibilidade, consiste em adicionar nós em locais importantes ou estratégicos do terreno virtual, tarefa que geralmente é feita manualmente pelo desenvolvedor, ou no caso de jogos, pelo *game designer*. Um ponto deste grafo deve ter pelo menos uma linha reta de visão para outro ponto incluso na estrutura.

Se o sistema para o qual a estrutura é utilizada restringe a movimentação dos agentes somente às arestas do grafo, a solução é uma boa escolha, contudo se o agente tem maior liberdade de movimento, algumas modificações devem ser feitas. Por exemplo, se para realizar uma busca de caminho os pontos inicial e final não fazem parte do grafo, os mesmos devem ser adicionados à estrutura e conectados a ela. Também, é possível buscar pelo nó da estrutura mais próximo ao ponto inicial, realizar um caminho até ele, e fazer uma busca de caminho pelo grafo até o nó mais próximo ao ponto de destino, dirigindo-se à este após completar o caminho pelo grafo.

Figura 2.2.2.2.1: Representação de mapa virtual utilizando grafo de visibilidade.



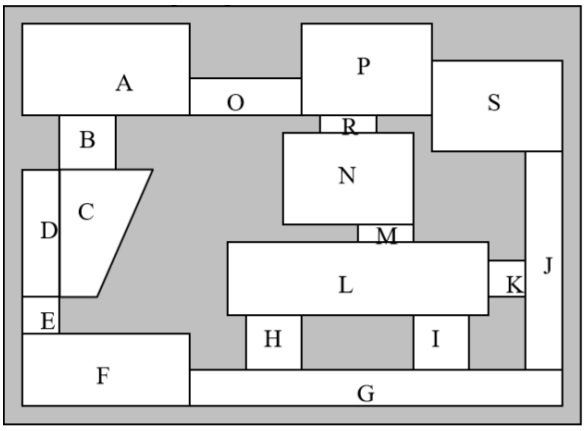
Fonte: Lima, E. S. **Waypoints e pathfinding – Inteligência artificial.** Disponível em: < edirlei.3dgb.com.br/aulas/ia_2012_1/IA_Aula_25_Waypoints_e_Pathfiding.pptx> Acesso em: 04 de jul de 2019.

2.2.2.3 Navigation Mesh

A estrutura de *Navigation Mesh* consiste em uma malha de polígonos convexos (a saber, polígonos simples nos quais nenhum segmento de reta conectando dois pontos de seu perímetro passa por fora de sua área), cujas áreas sejam trafegáveis pelo terreno virtual. Assim, cada nó do grafo representa um espaço convexo ao invés de um único ponto. Uma vez definidas as regiões de cada polígono, eles serão conectados para formar o grafo de navegação.

Com esta topologia, os polígonos podem descrever com maior detalhamento e precisão os obstáculos do terreno. Ainda, a estrutura fornece eficiência aos algoritmos de *pathfinding*, visto que nesta topologia estes operam sobre um número reduzido de nós.

Figura 2.2.2.3.1: Representação de mapa virtual utilizando navigation mesh.

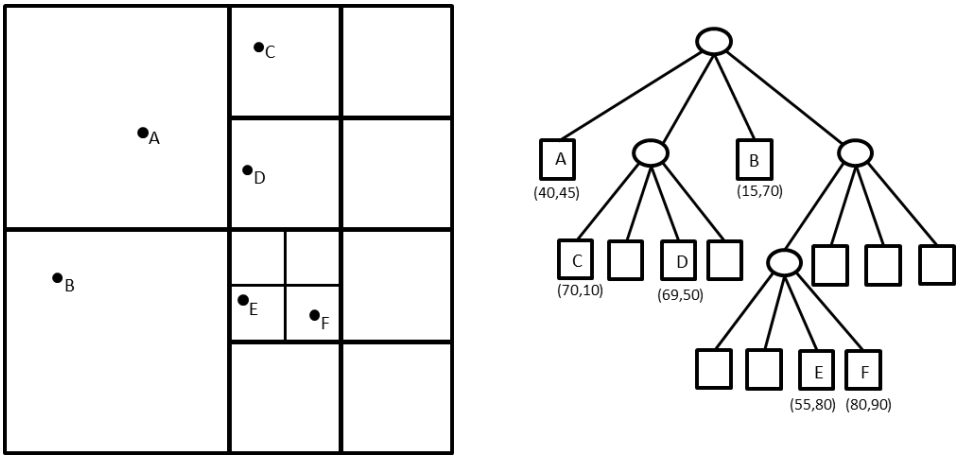


Fonte: Lima, E. S. **Waypoints e pathfinding – Inteligência artificial.** Disponível em: < edirlei.3dgb.com.br/aulas/ia_2012_1/IA_Aula_25_Waypoints_e_Pathfiding.pptx> Acesso em: 04 de jul de 2019.

2.2.2.4 Quadtree

Uma *quadtree* descreve uma classe de estruturas hierárquicas cuja propriedade em comum é o princípio da decomposição recursiva do espaço (Samet, 1984). Assim, a *quadtree* consiste em uma estrutura de dados em árvore na qual cada nodo interno tem exatamente quatro filhos, que, por sua vez, possuem ¼ do tamanho do nodo pai e cuja posição não sobrepõe a posição dos irmãos. Os nodos que não possuem filhos são chamados "nodos-folhas". A divisão dos nodos da estrutura ocorre baseada em critérios bem definidos e de acordo com os obstáculos presentes no terreno. A partir disso, os nodos podem ser classificados como transitáveis ou não.

Figura 2.2.2.4.1: Subdivisão de uma *quadtree*.



Fonte: OpenDSA. **The PR Quadtree.** Disponível em: <https://opendsa-server.cs.vt.edu/ODSA/Books/Everything/html/PRquadtree.html> Acesso em: 04 de jul de 2019.

Estruturas de dados hierárquicas são relevantes devido à sua capacidade de representar subconjuntos interessantes dos dados. Esse foco resulta em uma representação eficiente e tempos de execução aprimorados para algoritmos de *pathfinding*. Como discutido em (Brondani *et al.*, 2017) (Brondani *et al.*, 2018), a natureza hierárquica da representação utilizando *QuadTree* permite a otimização de algoritmos de *pathfinding*, uma vez que diferentes níveis de detalhe do terreno do mundo real são expressos na estrutura do mapa virtual resultante.

2.3 REPRESENTAÇÃO DE INFORMAÇÃO DE ALTURA EM MAPAS DE NAVEGAÇÃO

Para representar informações de altura em mapas de navegação, é necessário encontrar a estrutura de representação para o terreno virtual. A partir disso, é preciso escolher a forma de representação das informações referentes às características de cada região, que serão armazenadas em seus respectivos nodos. Neste trabalho, as informações relativas aos desníveis do terreno estão inclusas nos dados de interesse a serem armazenados na estrutura de representação do terreno virtual a ser utilizado, visto que o trabalho é voltado ao tratamento dessas características.

Na literatura (Ganganath *et al.*, 2015) (Chen *et al.*, 2009) (Pütz *et al.*, 2016), foram encontrados trabalhos que utilizam diferentes formas para representar a informação de altura na estrutura de representação de terrenos virtuais utilizada. Entre elas, é possível realizar a representação por meio da orientação da face dos nodos de uma estrutura (vetores normais), armazenamento de alturas máxima e mínima de cada região, grafos ponderados, entre outras.

Como descrito em (Ganganath *et al.*, 2015), para fazer o planejamento de caminho para movimentação de robôs foi utilizado um grafo ponderado. Este foi criado a partir de um Modelo de Elevação Digital de alta resolução (DEM), modelos úteis para representação precisa da elevação da superfície do terreno. Assim, cada nodo do grafo corresponde a um ponto da superfície do terreno. Dessa forma, ao realizar o *pathfinding* os pesos dos nodos, referentes às informações de relevo, são considerados.

Quando se trata de utilizar os vetores normais de um mapa para representar informações de altura, encontram-se diferentes formas de utilizar o recurso e convertê-lo em um valor de custo. Em (Chen *et al.*, 2009), a estrutura utilizada é *Navigation Mesh*, que como visto na seção 2.2.2.3, é formada por polígonos de diferentes formas, conectados. Cada um desses polígonos possui um vetor normal e todas as normais formam um mapa em gradiente, cuja variação significa a dificuldade de movimento entre dois polígonos. Na execução do algoritmo de busca de caminho a variação do gradiente é avaliada. Assim, o algoritmo irá retornar rotas cuja variação seja amena, representando menor dificuldade de movimentação em relação à inclinação entre dois polígonos.

Em (Pütz *et al.*, 2016), é abordada a ideia de *Navigation Mesh* em 3D para planejamento de caminhos em terrenos acidentados. A entrada principal para gerar a estrutura de navegação é uma malha triangular, construída a partir de algum procedimento de

reconstrução da superfície, gerando uma *NavMesh* sobre a qual são aplicados algoritmos de otimização de grafos. A estrutura resultante é combinada com dois grafos, um de vértices e um triangular, resultando em uma estrutura Graph Half Edge Mesh. Com esta, pode-se codificar várias características da superfície em várias camadas de custos de navegação para todas as arestas, que são fundidas em uma função de custo único para cada aresta. Entre estas camadas, encontra-se a que remete ao custo referente ao desnível ou diferenciação de altura local. Este custo também pode ser baseado no cálculo de normais, uma vez que essa estrutura permite associar vetores normais a cada triângulo e vértice da malha. Essa informação indica a inclinação local do terreno e pode ser incluída no cálculo de busca de caminho verificando a trafegabilidade de uma rota.

2.4 TÉCNICAS DE BUSCA DE CAMINHO QUE CONSIDERAM INFORMAÇÃO DE ALTURA

Apesar de diferentes formas de representar informações de altura ou inclinações de um terreno em um mapa de navegação, em geral, a informação é convertida em um custo a ser computado durante a busca de caminho. As formas de conversão dos dados de altura ou inclinação do terreno em custo também podem ser variadas. Neste caso, é preciso levar em consideração o problema a ser tratado pela busca: se é pretendido evitar tombamentos de agentes, se há intenção de menor gasto de energia, se envolve questões de logística, como gasto de combustível ou manutenção. Nesta seção, será apresentada a revisão de técnicas que utilizam a informação de altura representada em forma de custo no algoritmo de *pathfinding*.

No trabalho de (Ganganath *et al.*, 2015), o ângulo de inclinação do terreno influencia no consumo de energia do agente (ou robô, agente para o qual o artigo é voltado), influenciando na escolha do caminho. Levando em consideração as restrições do agente, é desenvolvida uma equação que demonstra o custo energético de travessia, a qual é aplicada à heurística do algoritmo de busca, gerando um balanceamento entre caminho mais curto e gasto de energia para realizá-lo, resultando sempre em caminhos factíveis aos agentes. Neste caso, foram incluídos pesos aos nodos que compõem a representação do terreno, além de informações sobre o atrito com o mesmo.

Em (Chen *et al.*, 2009), a dificuldade de passagem por cada nodo é representada pela inclinação obtida por meio de suas respectivas normais. O método que gera o custo leva em consideração três aspectos: o tamanho do caminho - busca sempre pelo menor caminho -, a

dificuldade de tráfego de acordo com a inclinação indicada pelas normais e as preferências/restrições referentes a cada tipo de agente. Assim, o custo é feito a partir da combinação linear das dificuldades de passagem de cada nodo.

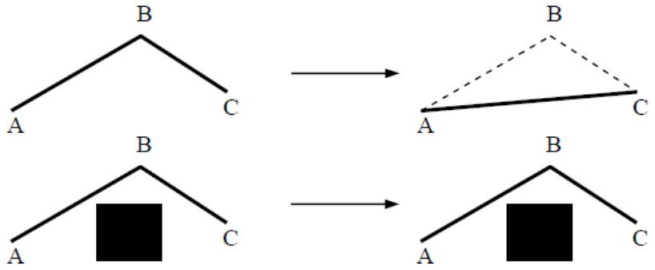
Como mencionado na seção anterior, os trabalhos de (Cheng and Tse, 2015) e (Pütz *et al.*, 2016) também utilizam os pesos obtidos a partir das informações disponíveis pela estrutura durante a busca de caminho. Neste trabalho, o custo foi elaborado em função das inclinações obtidas a partir das normais contidas na estrutura de representação do terreno, e o processo para chegar em sua definição é explicado na seção 3.

2.5 SUAVIZAÇÃO DE CAMINHOS

Os caminhos retornados pelos algoritmos de busca, em geral, fornecem uma trajetória que conecta pontos ou nodos consecutivos, gerando um caminho com muitos pontos, ou seja, segmentos relativamente curtos. Os caminhos encontrados pelo A*, por exemplo, são compostos por segmentos que conectam vizinho a vizinho, desde o ponto inicial até a posição objetivo, o que, por vezes, gera caminhos sinuosos, com pequenas curvas que poderiam ser evitadas. Essa característica pode aparentar falta de realismo durante a movimentação do agente quando comparada ao ambiente real, visto que o natural para se alcançar um destino é direcionar-se para atingi-lo objetivamente. Portanto, a fim de obter-se essa objetividade para atingir uma posição objetivo, algoritmos de suavização podem ser executados após o algoritmo de busca ter encontrado a rota entre os pontos informados, assim o caminho completo levaria o tempo do algoritmo de busca somado ao tempo do algoritmo de suavização.

Estes algoritmos eliminam pontos desnecessários ao caminho, desde que não haja obstáculos ou nodos não trafegáveis que necessitem ser evitados e, assim, impeçam que os pontos que formam as curvas sejam eliminados. Dessa forma, a distância a ser percorrida torna-se menor, fornecendo um caminho com maior qualidade. A Figura 2.5.1 ilustra a ideia de suavização ao eliminar pontos desnecessários ao caminho final, desde que não haja obstrução do caminho.

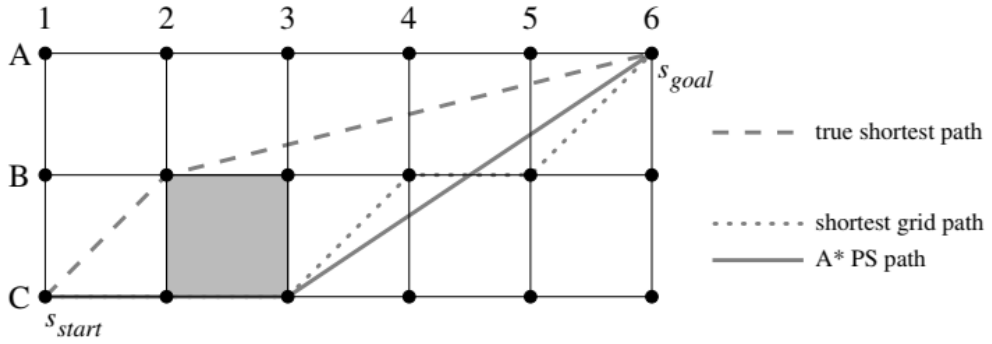
Figura 2.5.1: Processo que verifica se há campo de visão para eliminar vértices/nodos desnecessários ao caminho.



Fonte: Lima, E. S. **Waypoints e pathfinding – Inteligência artificial.** Disponível em: < edirlei.3dgb.com.br/aulas/ia_2012_1/IA_Aula_25_Waypoints_e_Pathfiding.pptx> Acesso em: 04 de jul de 2019.

Uma ideia de suavização pós-processamento do caminho retornado pelo algoritmo A* é utilizada em (Thorpe e Matthies, 1984). Naturalmente, o comprimento do caminho fica menor, enquanto o tempo total de execução é acrescido. O pós-processamento geralmente encontra o caminho mais curto possível na estrutura de representação, entretanto isso não significa que seja o menor caminho real (Figura 2.5.2). Ainda assim, o caminho com a suavização se torna menor e mais próximo do realismo. A Figura 2.5.2 exemplifica uma situação na qual apresenta a comparação entre o menor caminho real, o caminho encontrado utilizando cada nodo do grid a cada passo do algoritmo de busca e o menor caminho encontrado pelo pós-processamento que aplica suavização, após encontrar o caminho utilizando os nodos do grid.

Figura 2.5.2: Comparação entre o menor caminho real, o caminho encontrado considerando o grid (A*) e o caminho associado ao pós-*smoothing*.



Fonte: (Nash *et al.*, 2007).

Um modelo de algoritmo de suavização simples (Algoritmo 2) e funcional, o qual produzi bons resultados, é proposto por (Botea *et al.*, 2004). Verifica-se quais pontos ou nodos do caminho podem ser retirados, substituindo a conexão por uma linha reta. O processo inicia de um lado da solução. Para cada nó na solução, é verificada a possibilidade de alcançar um nó subsequente no caminho em uma linha reta. Se isso acontecer, então o caminho linear entre os dois nós substitui a sequência sub-ótima inicial entre esses nós. Como mostrado no Algoritmo 2, a entrada do algoritmo de pós-processamento é o caminho encontrado pelo algoritmo de busca utilizado. Inicia-se um contador em 0, neste caso nomeado por *k*, e o nodo inicial do caminho recebido por parâmetro é atribuído como nodo de entrada do novo caminho com suavização. Após esta etapa inicial, um laço é executado a fim de verificar a possibilidade de traçar o caminho diretamente ao próximo ponto, eliminando nodos desnecessários. Isto pode ser observado nas linhas 34 à 36. Caso não tenha visão para o próximo nodo, ou seja, o caminho é obstruído e não pode ser realizado diretamente ao próximo nodo omitindo o nodo atual no laço, o contador de nodos do novo caminho é atualizado, sendo incrementado (um novo nodo é adicionado ao caminho suavizado). Assim, o nodo no índice do contador recebe o nodo do caminho de entrada que está sendo verificado no laço, neste caso *Si*. Uma vez que o laço de repetição verifica nodos do ponto inicial do caminho de entrada até o penúltimo nodo, ao final, obrigatoriamente é incrementado o contador do caminho suavizado e o nodo final do caminho de entrada é atribuído ao nodo final do caminho suavizado que está sendo buscado. Dessa forma, garante-se que o nodo final é atingido e o caminho pós-processado é retornado.

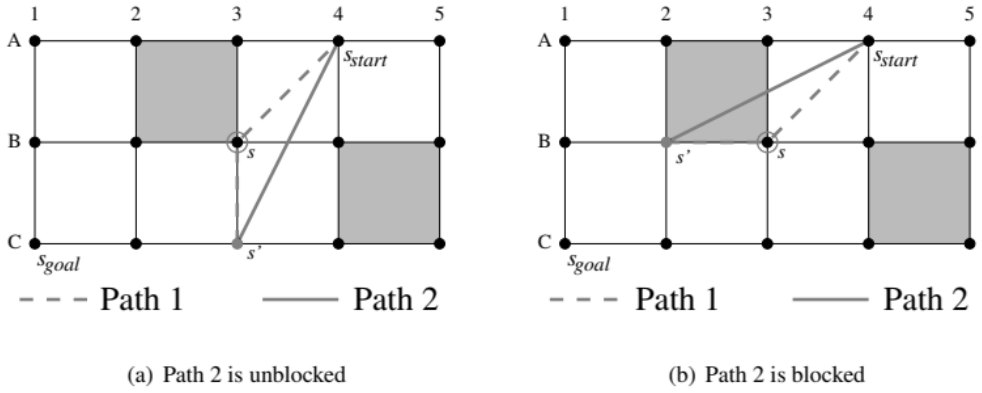
Algoritmo 2: Algoritmo básico para aplicar *smooth* pós busca de caminho.

```
30 PostSmoothPath([s_0, ..., s_n])
31     k := 0;
32     t_k := s_0;
33     foreach i := 1 ... n − 1 do
34         if NOT LineOfSight(t_k, s_{i+1}) then
35             k := k + 1;
36             t_k := s_i;
37     k := k + 1;
38     t_k := s_n;
39     return [t_0, ..., t_k];
40 end
```

Fonte: (Nash *et al.*, 2007).

Buscando uma alternativa que possibilite a suavização e também permita fazer o tratamento de inclinações do terreno, encontra-se o algoritmo Theta* (Nash *et al.*, 2007). Tendo sido desenvolvido a partir do A*, o Theta* é uma variação deste na qual ao expandir o caminho para o próximo vizinho, faz uma verificação se é possível traçar uma rota do pai do nodo atual até o vizinho para o qual o algoritmo pretende expandir o caminho. Basicamente, o Theta* combina a ideia de busca de caminhos do A* em grafos de visibilidade com a ideia de A* em estruturas de grid. Especificando sua diferença em relação ao A*, ao atualizar o custo g e o pai de um vizinho visível não expandido dos nodos, ele considera duas possibilidades de continuar o caminho. Uma das opções é o caminho feito pelo A*, que considera o caminho do nodo inicial até o nodo atual (s) verificado e a distância em linha reta para seu vizinho (s′) $[= \text{dist}(s, s')]$. A segunda opção de caminho considera o caminho do nodo inicial até o pai do nodo atual e verifica se é possível formar um caminho do pai para o vizinho visível $[= \text{dist}(\text{parent}(s), s')]$, verificando a distância em linha reta. A segunda opção é possível quando existe o campo de visão do pai do nodo atual para o vizinho visível. Ter o campo de visão significa não ter algum nodo obstruído entre nó observador (pai do nodo atual) e observado (vizinho visível), que impeça a passagem do nodo pai ao vizinho. Na Figura 2.5.3 é possível visualizar esta ideia. Se não há obstrução, o caminho 2 é realizado, caso contrário, o caminho 1 é realizado. A diferença também pode ser verificada no pseudocódigo detalhado no Algoritmo 3.

Figura 2.5.3: Comparação das duas opções de caminho que o Theta* pode realizar.



Fonte: (Nash *et al.*, 2007).

Algoritmo 3: UpdateVertex do Theta* - atualização do custo g e decisão do próximo nodo a ser atingido.

```
41 UpdateVertex(s,s')
42     if LineOfSight(parent(s), s') then
43         /* Path 2 */
44         if g(parent(s)) + c(parent(s), s') < g(s') then
45             g(s') := g(parent(s)) + c(parent(s), s');
46             parent(s') := parent(s);
47             if s' ∈ open then
48                 open.Remove(s');
49             open.Insert(s', g(s') + h(s'));
50     else
51         /* Path 1 */
52         if g(s) + c(s, s') < g(s') then
53             g(s') := g(s) + c(s, s');
54             parent(s') := s;
55             if s' ∈ open then
56                 open.Remove(s');
57             open.Insert(s', g(s') + h(s'));
58 end
```

Fonte: (Nash *et al.*, 2007).

O algoritmo demonstra eficiência ao aplicar a ideia de suavização, uma vez que já planeja o caminho de forma suavizada, sem necessidade de pós-processamento. Ainda, permite fazer a busca baseada no tratamento de restrições, transcendendo a ideia de apenas verificar se os nodos são trafegáveis ou não. Esta ideia favorece este trabalho, que visa tratar inclinações ao passo que busca um caminho suavizado. Em geral, tais características podem ser usadas como indicadores de qualidade do caminho resultante do algoritmo.

2.6 ALGORITMOS DE PATHFINDING HIERÁRQUICOS

Estruturas de representações mais detalhadas descrevem com maior precisão as características do terreno. Contudo, com nodos menores, cria-se um espaço de busca muito grande, principalmente quando se trata de grandes terrenos, como em um ambiente de simulação. Assim, essa representação tem alta resolução. Por outro lado, uma representação menos detalhada, como nodos maiores, gera um espaço de busca mais simples. Todavia não oferece precisão na descrição das características do terreno representadas no terreno virtual,

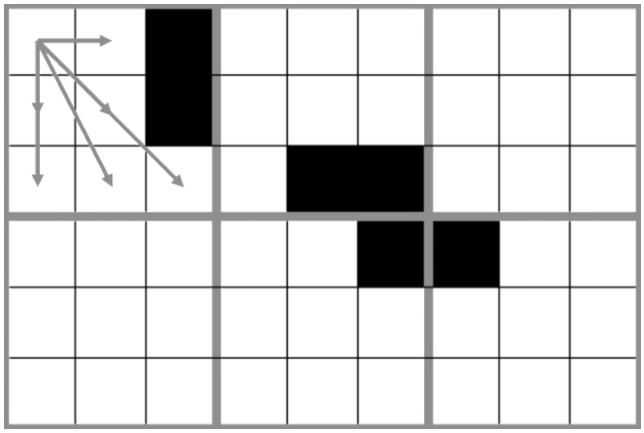
tendo, portanto, baixa resolução. Dessa forma, o espaço de busca mais simples permite que o algoritmo de busca execute de forma mais rápida em relação a um espaço de busca mais complexo.

A abordagem hierárquica é uma proposta que visa reduzir a complexidade dos problemas de busca de caminho. A técnica consiste em abstrair um mapa de navegação em clusters locais (Botea *et al.*, 2004), como se uma grade fosse adicionada sobre o mapa, a fim de simplificar o processo de busca.

(Botea *et al.*, 2004) propõe o Hierarquical Path-Finding A* (HPA*), sugerindo um ou mais níveis de abstração com o intuito de reduzir a complexidade do espaço de busca sem necessariamente perder a descrição detalhada do terreno. Para isso, o mapa do terreno é abstraído em clusters pré-computados, os quais agrupam nodos-folha. Assim, para realizar a busca, é feita uma busca abstrata para encontrar o caminho mais "grosso", entre os grandes clusters. Em seguida é realizado o que pode ser chamado de caminho concreto, que é uma busca de caminho refinada no interior de cada cluster. Como descrito em (Botea *et al.*, 2004), experimentos demostram que esta técnica reduz o custo computacional da busca e produz caminhos resultantes que estão dentro de 1% do caminho ideal. Essa abordagem hierárquica pode ser explorada em conjunto com diferentes topologias. Entretanto, para isso é necessário que os clusters sejam pré-computados.

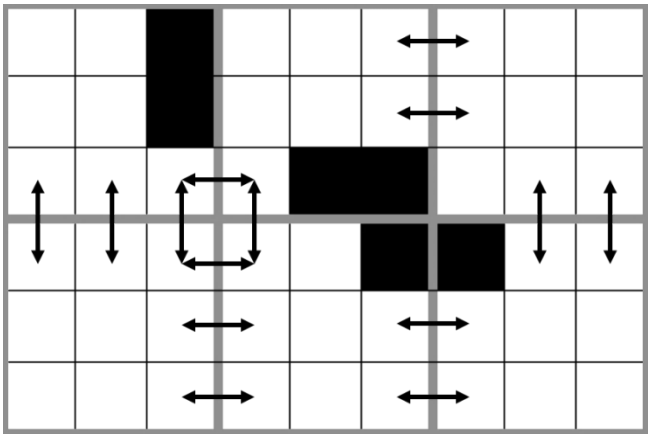
A ideia de pré-processamento pode ser visualizada pelas imagens apresentadas nas Figuras 2.6.1, 2.6.2 e 2.6.3. Tendo os clusters definidos, é necessário pré-processar os caminhos dentro de cada grade, todos os transitáveis contra todos transitáveis, como mostrado na Figura 2.6.1. Uma vez que os valores internos a cada cluster sejam armazenados, são detectados os nodos de fronteiras, que os conectam (Firgura 2.6.2).

Figura 2.6.1: Nodos internos do cluster buscando caminho de fronteira a fronteira.



Fonte: Autor.

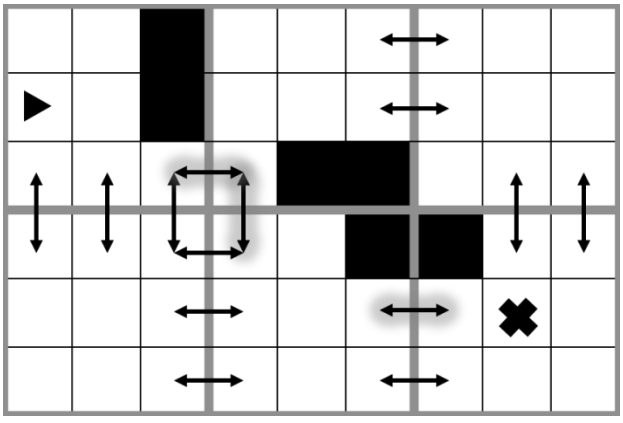
Figura 2.6.2: Conexão dos clusters pelos nodos de fronteira.



Fonte: Autor.

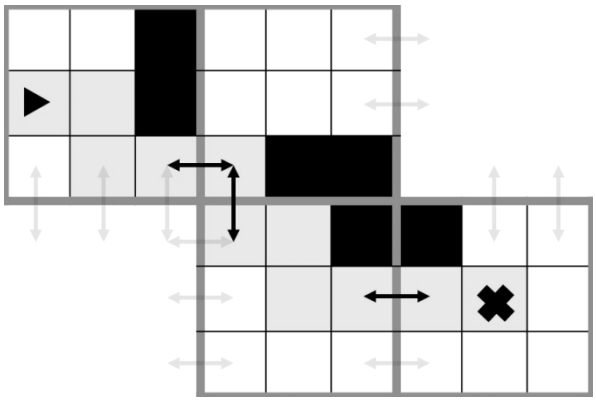
Ao executar o algoritmo, durante a execução do sistema, são descobertos os clusters iniciais e finais, de acordo com os pontos informados, visto que estes estão respectivamente contidos em seus clusters, e então são descobertas todas as ligações que se deve percorrer, de cluster a cluster (Figura 2.6.3). Ao descobrir os clusters relevantes para o caminho, o algoritmo padrão de busca é executado internamente a cada um, formando o caminho (Figura 2.6.4). Uma vez que tais informações computadas off-line (informações pré-processadas e armazenadas em uma memória) são usadas, o tempo de busca diminui quando comparado ao tempo de executar o algoritmo de *pathfinding* somente em tempo de execução, ponto a ponto.

Figura 2.6.3: Conexão dos clusters encontrando nodos vizinhos de fronteira.



Fonte: Autor.

Figura 2.6.4: Buscando o caminho após encontrar os clusters relevantes para o mesmo.



Fonte: Autor.

O algoritmo hierárquico proposto em (Botea *et al.*, 2004) resolve os problemas referentes ao tempo de execução e memória, contudo não resolve os problemas de restrições da grade. (Van Elswijk *et al.*, 2013) desenvolve a ideia do Hierarquical Path-Finding Theta* (HPT*), combinando a técnica hierárquica do HPA* com o algoritmo de busca Theta*. Os experimentos do trabalho concluem que o HPT* combina os benefícios de ambos os algoritmos de forma bem sucedida, que seriam: 1) a qualidade do caminho fornecido pelo Theta* e 2) os benefícios em relação ao tempo médio de computação e requisitos de memória do HPA*. Os procedimentos para pré-processamento e busca de caminho são muito semelhantes ao HPA*, com exceção do fato de 1) no pré-processamento, os nós internos do cluster serão conectados se houver um caminho encontrado pelo Theta* entre eles e 2) ao realizar a busca de caminho concreta, em tempo de execução, esta será realizada utilizando o Theta*.

A ideia proposta em (Van Elswijk *et al.*, 2013), foi utilizada de embasamento para a implementação objetivo deste trabalho: desenvolver um algoritmo que utilize técnica de *smooth* (Theta*) a fim de fornecer maior realismo à navegação e que também ofereça tempo de execução e uso de memória adequados (HPA*), além de tratar o problema de alturas e inclinações.

3 ESTRUTURA DE REPRESENTAÇÃO UTILIZADA

Neste trabalho, uma estrutura de *quadtree* hierárquica e irregular é explorada na representação do mapa de navegação do ambiente de terreno virtual. O nível mais alto na hierarquia representa a área total do mapa do terreno. Na hierarquia, quanto mais profundo o nodo, maior o nível de detalhe da respectiva região do terreno virtual.

No sistema de simulação no qual este trabalho está inserido, a subdivisão da *quadtree* durante a construção do mapa de navegação considera diferentes características do terreno, como, por exemplo: a densidade da vegetação, as formas da paisagem do terreno e quão íngreme ela é (alturas e inclinações, aspecto de interesse neste trabalho), e a presença de rios ou correntes de água. Caso uma das características de terreno relevantes para fins de simulação seja identificada no mapa do mundo real usado (visto que terrenos virtuais são construídos usando informações de mapas GIS de regiões reais), um novo nível na hierarquia da *quadtree* é construído adequadamente. Este processo é repetido até que uma condição final de subdivisão da hierarquia da *quadtree* seja satisfeita, a qual é definida de acordo com as características do terreno analisadas ou atingindo o nível máximo da hierarquia da *quadtree*. No final, a representação irregular do terreno virtual é expressa pelos nodos-folha da estrutura de representação hierárquica.

Na execução dos algoritmos de busca de caminho utilizando essa representação, a aplicação da hierarquia é bem-sucedida, uma vez que o processo de pesquisa hierárquica permite diminuir o custo geral da pesquisa, pois o número de nós visitados no mapa de navegação é reduzido. Como descrito nos trabalhos de (Botea *et al.*, 2004) (Botea *et al.*, 2013), (Pelechano e Fuentes, 2016) e (Algfoor *et al.*, 2015), os nodos visitados estão diretamente relacionados ao tempo de computação da busca.

Mais especificamente, a *quadtree* utilizada pode ser definida como:

Definição 1: "rooted tree" é um grafo direcionado no qual:

- Uma "rooted tree" consiste em uma tupla:
  - $Tupla < maxLevel, root >$
- Cada aresta na árvore é chamado de nodo (n). Um nodo n consiste na tupla:
  - $n < parent, children, level, isLeaf, refined >$
- Um nodo n que gera novos nodos (n1) é chamado nodo pai dos nodos gerados:
  - $n = n1.parent$

- Um nodo (n1) gerado por um nodo pai (n) é chamado de "nodo filho". Um nodo filho (n1) é parte de um conjunto de nodos (n.*children*) consistindo dos nodos gerados por n;
    - $n.children = \{n1 \mid n = n1.parent\}$
- O nodo raiz (*Tree.root*) da árvore não possui um nodo pai:
    - $Tree.root.parent = null$
- Há sempre um caminho da raiz para qualquer nodo na árvore;
- O campo $n.level$ indica o nível do nodo n na hierarquia da árvore;
- $n.refined$ é um atributo booleano que indica que novos nodos temporários foram gerados a partir deste nodo n;
- O atributo $Tree.maxLevel$ indica o nível hierárquico máximo da árvore.

Definição 2: (*Navigation QuadTree*) Uma *QuadTree* é uma "rooted tree" na qual cada nodo tem exatamente zero ou quatro filhos nodos. Enquanto *QuadTree* de navegação, consiste em uma estrutura hierárquica para representar os mapas de navegação. Essa estrutura gera uma grade irregular expressa nos nodos folha da representação hierárquica.

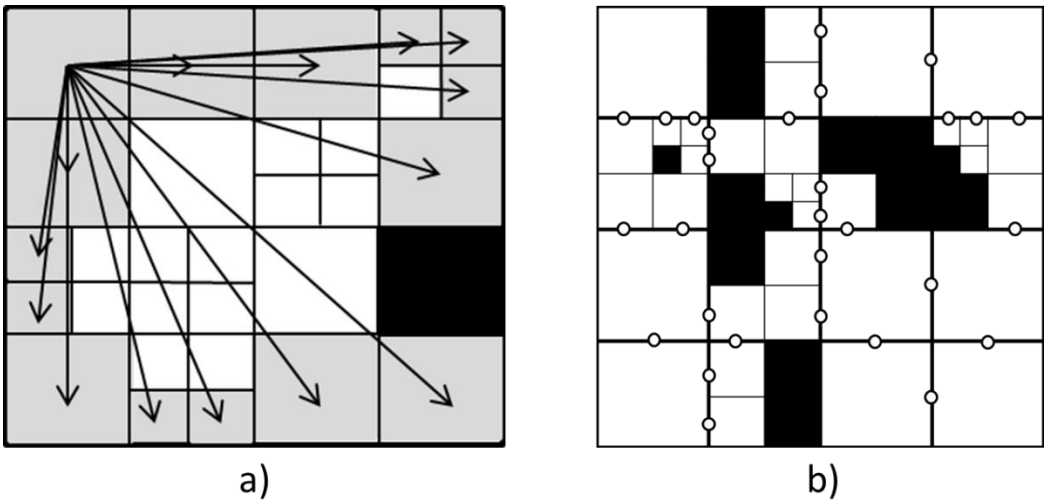
Para gerar uma grade irregular, cada nodo possui as seguintes informações:

- Atributos representando informações do terreno virtual:
    - $n < minHeight, maxHeight, neighbors, hasRiver, walkable, normalNorth, normalSouth, normalEast, normalWest >$
- Os atributos $n.minHeight$ e $n.maxHeight$ fornecem informações sobre altura mínima e máxima, encontradas na região do terreno do mundo real representada pelo nodo;
- O atributo $n.\mathrm{neighbors}$ consiste em uma matriz representando os nodos sob consideração. Baseado na grade de navegação construída, este atributo expressa a conexão entre os nodos, possibilitando a travessia pelos nodos da grade;
- O atributo $walkable$ indica se o nodo está acessível ou não por agentes inseridos nas simulações. Com base nas informações das características do terreno representadas por esse nodo, ele pode ser definido como transitável ou não;

- Os atributos $normalNorth, normalSouth, normalEast$ e $normalWest$ são os quatro vetores normais do nodo, referentes a cada direção deste, os quais fornecem informações sobre o ângulo de inclinação quando comparados ao vetor *Up* (vetor normal a uma superfície plana, apontando para cima, cujas coordenadas no espaço são (0, 1, 0)).

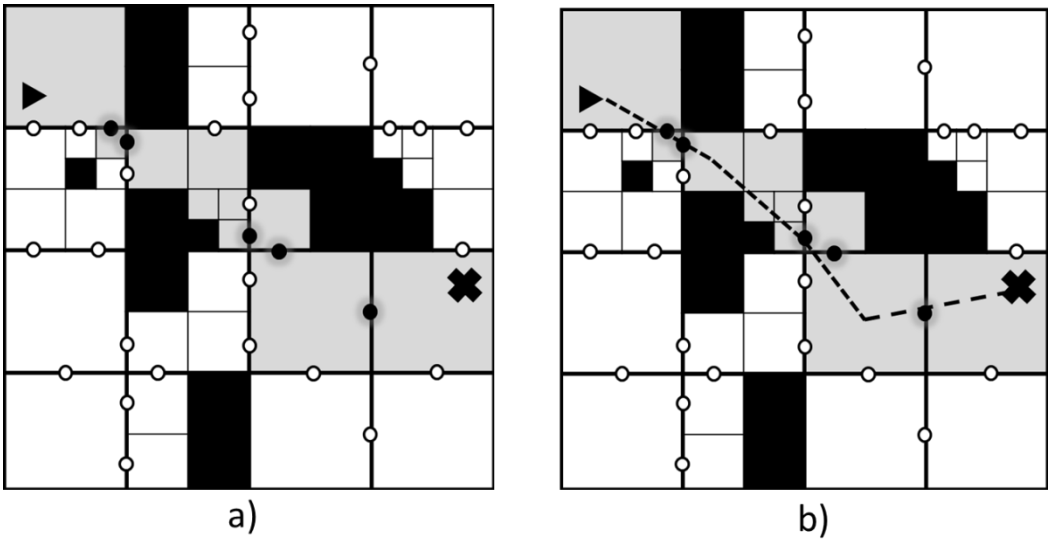
Para fornecer a hierarquia durante o processo de busca de caminho utilizando a estrutura irregular, um nível mais alto da *quadtree* (a saber, utilizamos o nível 6 da hierarquia) é utilizado para formar os clusters, ideia explanada na seção 2.6. A partir da formação dos clusters, o processo para descobrir nodos de fronteira e processar cada cluster pode ser feito assim que o simulador inicia. Caso ocorra alguma mudança na estrutura de navegação, o processo é repetido durante a execução, sendo este um processo online. Parte do processamento do cluster envolve isolar os nodos de fronteira (Figura 3.1, item b) e buscar as distâncias de todas as fronteiras contra todas as fronteiras do cluster (Figura 3.1, item a). Ao executar um algoritmo de busca, este irá inicialmente planejar o caminho entre clusters (Figura 3.2, item a), pelas ligações entre fronteiras, processo que é denominado de busca do caminho abstrato. Em seguida, utiliza o algoritmo escolhido para busca de caminho internamente a cada cluster, processo nomeado de busca do caminho concreto. Em cada cluster, é realizada uma busca de caminho entre dois nodos de fronteira selecionados, para entrada e saída do cluster. A união dos caminhos a cada cluster, forma o caminho final (Figura 3.2, item b). Ainda, os dados utilizados em tempo de busca são salvos como um cache em arquivo, que pode ser recriado a qualquer momento. Por exemplo, os nodos de fronteira podem ser armazenados em uma lista para cada nodo cluster, a fim de reduzir o tempo de busca.

Figura 3.1: a) Distância armazenada de todos os vizinhos de fronteira contra os demais, internamente a cada cluster; b) Identificação dos nodos de fronteira de cada cluster.



Fonte: Autor.

Figura 3.2: a) Caminho abstrato, isto é, clusters que serão utilizados para busca do caminho concreto; b) Caminho concreto, com execução do algoritmo a cada cluster.



Fonte: Autor.

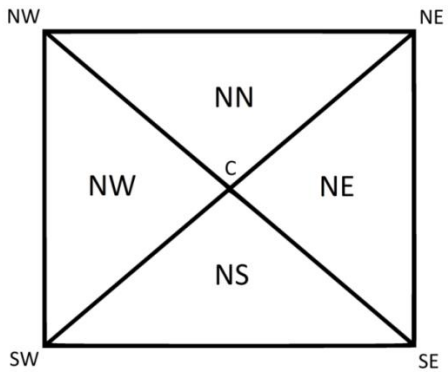
Na estrutura utilizada, foram inseridas tanto as informações de altura máxima e mínima de cada nodo quanto suas normais, para que posteriormente pudessem ser utilizadas pelos algoritmos de navegação, a fim de obter as inclinações de cada nodo a partir desses atributos de representação de terreno. Como o algoritmo desenvolvido pretende tratar a

dificuldade de movimento, decidiu-se que a forma ideal de representação do relevo para ser tratada seria o uso de inclinações. Uma inclinação pode ser considerada uma informação impeditiva para a movimentação e, assim, mais relevante para a computação de caminhos mais seguros.

Neste trabalho, os atributos de maior relevância referem-se à obstrução do nodo ou não ($walkable$) e os atributos que fornecem informações sobre as inclinações do terreno. Neste caso, inclinações são representadas por vetores normais às quatro faces direcionadas (norte, sul, leste, oeste) que compõem o nodo ($normalNorth$, $normalSouth$, $normalEast$ e $normalWest$). Estes contribuem na elaboração dos custos a serem computados pelo algoritmo de *pathfinding*.

As normais são construídas a partir do produto vetorial dos vetores que compõem cada região triangular, internos aos nodos-folha da *quadtree* (Figura 2.3.1), que referem-se às respectivas direções destes. Este processo utiliza dois pontos dos cantos do nodo, relativos à direção da normal que está sendo estabelecida (norte, sul, leste, oeste), e o ponto central do mesmo. Dados estes pontos, os vetores que formam a região triangular são obtidos, possibilitando a realização do produto vetorial, que por sua vez gera um vetor ortogonal aos dois vetores. Visto que o vetor é perpendicular à face da região triangular, este será o vetor normal. Assim foram definidas: normalNorth (NN), normalSouth (NS), normalEast (NE), normalWest (NW). Obtendo um vetor normal para cada direção do nodo, é possível utilizar no custo de movimento apenas os valores de inclinação pelas quais o agente irá passar. Assim, tem-se a ideia de custo de entrada em um nodo, de acordo com a direção pela qual o mesmo é atingido, e ideia do custo para saída de um nodo, custo para sair do centro do nodo atual até a fronteira com um vizinho.

Figura 3.3: Nodo da *quadtree* com regiões triangulares internas, com suas respectivas normais.



Fonte: Autor.

O cálculo das normais é obtido da seguinte forma:

$$\begin{cases} NS = (SE - C) * (SW - C) \\ NW = (SW - C) * (NW - C) \\ NN = (NW - C) * (NE - C) \\ NE = (NE - C) * (SE - C) \end{cases}$$

Tendo a representação das inclinações das quatro direções do nodo, obtém-se informação de altura para entrada e saída do mesmo, além de fornecer maior detalhamento na representação das informações do terreno.

4 O CUSTO ABORDADO NA BUSCA DE CAMINHO AO CONSIDERAR ALTURAS

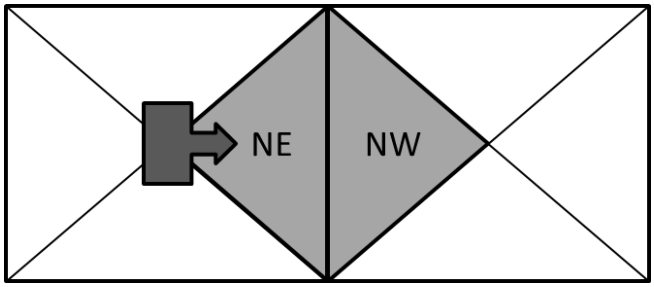
A fim de tratar o problema de elevações no mapa virtual do terreno durante a busca de caminho e de garantir um caminho cujo relevo seja adequado ao tráfego de agentes, é necessário adicionar um custo ao algoritmo de *pathfinding*. Dessa forma, foi necessário escolher uma forma de utilizar as informações disponíveis na estrutura de representação do mapa virtual de navegação para gerar um custo a ser utilizado na busca de caminho. Para isto, foram pesquisadas abordagens de custo para tratamento de alturas. Como mencionado na seção 2.3, diferentes formas podem ser utilizadas para converter os dados disponíveis em custos, sendo a escolha baseada no problema que se pretende resolver.

Como foi utilizada a abordagem de angulação das inclinações do relevo, obtidas através das normais de cada nó triangular da *quadtree*, utilizou-se o cosseno do ângulo como um valor de peso. O peso é o que mais fornece influência na decisão de escolha do caminho, portanto onde houvesse maior peso, aplicar-se-ia um menor custo. Dessa forma, quanto menor o ângulo de inclinação (θ), maior é seu cosseno, oferecendo maior peso. Logo, fazendo a inversa do cosseno, obtém-se um menor custo. Assim, foi utilizado custo = $\frac{1}{\cos(\theta)} * \beta$, no qual β é um valor de influência do custo na escolha do trajeto (quanto maior seu valor, maior a tendência a desviar de inclinações no trajeto).

Como cada nodo folha da *quadtree* é formado por quatro regiões triangulares que remetem às direções do nodo, cada nodo folha possui quatro normais. Estas foram definidas baseadas nas direções de cada nodo, permitindo tratamento de inclinação tanto na direção de entrada quanto na direção de saída de cada um. Para determinar o custo de passagem pelo nodo, foram selecionadas as inclinações na direção de saída do nodo atual e na direção de entrada do nodo vizinho, partes do nodo pela qual o agente transita durante seu movimento. Por exemplo, se o agente está se movimentando direcionado a leste, ele realiza saída de seu nodo atual passando pelo seu lado leste, utilizando a respectiva normal. Dessa forma, ele realiza entrada no nodo vizinho pela fronteira oeste, utilizando o vetor normal deste lado do nodo para o qual a direção do movimento ocorre (Figura 4.1). Apesar de utilizar as informações de altura de acordo com as direções do nodo pelas quais o agente irá transitar, é indispensável a verificação dos demais vetores normais do nodo, visando garantir se há ou não uma inclinação acentuada neste, situação que o tornaria intransitável. Caso o nodo possua

uma inclinação acima do permitido para a movimentação segura do agente, o nodo é ignorado pelo algoritmo, não passando pelo restante do processo de verificação.

Figura 4.1: Exemplo de vetores normais de entrada e saída dos nodos, baseando-se na direção do movimento.



Fonte: Autor.

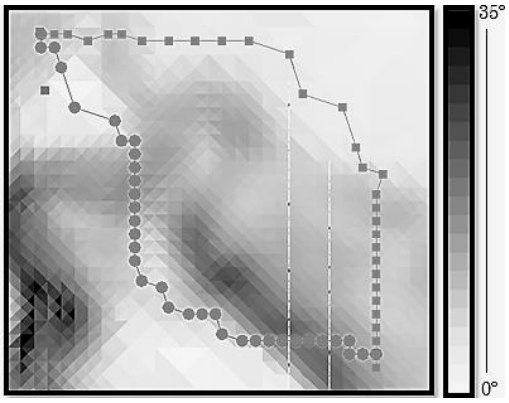
Tendo as inclinações de entrada e saída, é feita uma verificação se estão concentradas em um intervalo entre 5º e α, sendo α um valor máximo de inclinação que permite travessia, podendo este valor ser parametrizável. Caso esteja no intervalo, o custo é atribuído conforme o cálculo descrito anteriormente. Estando abaixo deste valor, o custo permanece 0, pois pretende-se tratar apenas inclinações relevantes à movimentação. Neste trabalho, foi considerado serem relevantes inclinações a partir de 5º devido à definição de montanha fornecida pela ONU (Blyth, 2002). Assim, o custo pode ser resumido como:

$$\begin{cases} \theta < 5º, mScost = 0 \\ 5º \leq \theta < \alpha, mScost = \dfrac{1}{\cos(\theta)} * \beta \end{cases}$$

Após escolha do custo, a eficácia do processo de busca foi avaliada visualmente, selecionando pontos iniciais e finais de forma manual no sistema de simulação. A utilização do custo foi aplicada ao algoritmo já utilizado pelo sistema de simulação, o algoritmo de busca de caminhos A*. O algoritmo até então utilizado no simulador, não considerava as elevações do relevo, utilizando apenas a menor distância entre pontos iniciais e finais das rotas como finalidade. Assim, o algoritmo ignorava trechos possivelmente inadequados ao tráfego de agentes inseridos no mapa de navegação. O único tratamento oferecido para a questão de altura era bloquear nodos a partir de determinada inclinação, tratamento mais simplório utilizado para resolver problemas de obstáculos ou indicativos de dificuldades/riscos em terrenos virtuais.

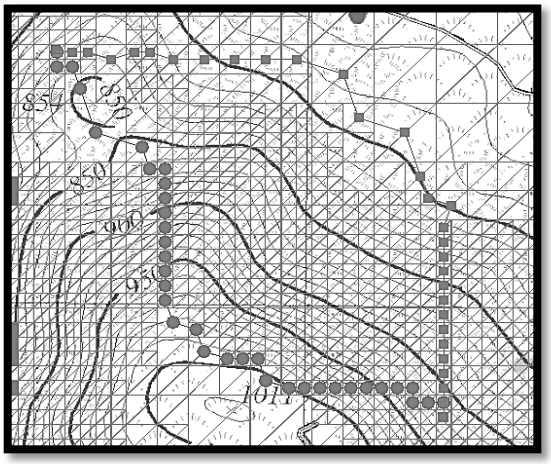
Incrementando o algoritmo de forma a considerar os desníveis do terreno, tratando problemas que pudessem interferir negativamente na movimentação de agentes, as rotas se tornaram menos custosas e mais realistas, contornando locais específicos que poderiam ser empecilhos na trajetória, como demonstrado no exemplo apresentado nas Figuras 3.1 e 3.2. Nestas figuras, percebe-se que o algoritmo buscou um trajeto por regiões cujas inclinações eram amenas quando comparadas às regiões inclusas na rota encontrada pelo algoritmo A* padrão. O mapa de vetores normais indica as inclinações destas de forma visual, visto que quanto mais escura a região, maior sua inclinação. No exemplo da imagem, havia sido determinado 35º como valor de inclinação limite para tráfego.

Figura 4.2: Caminhos encontrados pelo A* padrão (círculos) e pelo A* tratando inclinações (quadrados), exibidos no mapa de normais, cujo gradiente indica a variação nas inclinações.



Fonte: Autor.

Figura 4.3: Região do mapa virtual do simulador, com demarcação dos nodos.



Fonte: Autor.

Verificando o êxito da aplicação do custo ao algoritmo A*, ao refletir bons resultados na busca de caminho, foi possível evoluir a implementação para uma abordagem que utilize a aplicação de suavização ao caminho A ideia de suavização foi experimentada utilizando o algoritmo Theta*, que busca um caminho ao passo em que o suaviza.

5 HPATHETA*: O ALGORITMO PROPOSTO

Percebendo a funcionalidade do custo ao aplicá-lo ao algoritmo A*, e comparando a implementação deste com tratamento de alturas em relação à sua versão padrão, foi testada a aplicação do custo em um algoritmo que proporcione a suavização de caminhos, o Theta*. A técnica de suavização tende a tornar o caminho mais realista, melhorando sua qualidade quando se trata de menor distância a ser percorrida. Contudo quando trabalhado com alturas, a técnica padrão para suavização desconsidera quais regiões poderiam ser melhores para passagem, visando tanto menor distância quanto menor custo.

Compreendendo melhor como o Theta* realiza a busca e sua técnica característica (o campo de visão), foi necessário planejar uma forma de incluir o tratamento de inclinações neste algoritmo. Neste caso, apenas verificar o custo de inclinação da posição atual para o vizinho a ser expandido não é suficiente, pois é preciso tratar o problema de alturas no terreno ao utilizar a técnica do campo de visão. Na literatura não foram encontrados trabalhos que abordem tratamento do relevo aliado ao Theta* ou aos algoritmos de suavização.

A solução encontrada envolve:

1) obstruir o campo de visão quando há em seu caminho alguma inclinação igual ou maior que α, a inclinação máxima permitida para passagem;

2) realizar uma soma dos custos adquiridos pelas inclinações de entrada e saída dos nodos encontrados no campo de visão de um nodo ao outro, atribuindo dessa forma um custo ao campo de visão;

3) caso o campo de visão esteja obstruído, realizar o caminho direcionado ao vizinho consecutivo, como ocorre no algoritmo A*, verificando o custo do movimento ao próximo nodo.

O campo de visão, nomeado propositalmente neste trabalho como “Campo de visão míope” (Algoritmo 5), foi projetado para:

1) não ser obstruído desde que não tenha bloqueio de nodo ou alguma inclinação que forneça risco e impossibilite o movimento;

2) para que a soma das inclinações ao longo do seu campo indiquem se o trajeto indicado pelo campo de visão acarretará dificuldades para ser realizado.

Por isso a denominação “míope”, pois é possível ter campo de visão livre enquanto o custo total das inclinações de seu campo não for muito elevado em relação às demais possibilidades de trajeto e enquanto não impactar negativamente no caminho.

Algoritmo 4: Algoritmo que seleciona o caminho a ser tomado de acordo com obstrução ou não do campo de visão "míope".

```
59  UpdateVertex (s, s', maxAng)
60    If s.pathParent != null && CanSeeMyopic(s.pathParent, s', out slopeCost) then
61        doPathTwo(s, s', slopeCost)
62    else
63        doPathOne(s, s', maxAng)
64  end
```

Fonte: Autor.

Algoritmo 5: Campo de visão modificado para tratar altura.

```
110. CanSeeMyopic(fromNode, toNode, out slopeCost, maxAngle)
111.     while openSightPath.Count > 0 do
112.         if Midway = toNode then
113.             observedReached = true
114.             break;
115.         if !Midway.walkable || Midway.maxAngle >= maxAngle then
116.             pathObstructed = true
117.             break;
118.         cost = switch (s'.cellDirection)
119.         sumSlopes += cost
120.     slopeCost = sumSlopes
121. end
```

Fonte: Autor.

Com este tratamento, o rumo do caminho sendo analisado pode mudar de acordo com os custos fornecidos ao verificar cada campo de visão. O tratamento refere-se ao chamado *"Path 2"* (Algoritmo 6), mencionado e apresentado em pseudocódigo na seção 2.5, quando descrito o processo de busca do algoritmo Theta*. Se o "campo de visão míope", o qual é um campo de visão ponderado, não está obstruído, retorna um custo a ser computado e utilizado na tomada de decisão da rota pelo algoritmo.

Algoritmo 6: Algoritmo que realiza o "*Path 2*" tomado pelo Theta*, modificado para receber o custo de inclinações fornecido pelo campo de visão.

```
65  doPathTwo(s, s', slopeCost)
66  |  newCostToMove = s.pathParent.Gcost + c(s.pathParent, s') + slopeCost
67  |  if newCostToMove < s'.Gcost then
68  |  |  s'.Gcost = newCostToMove
69  |  |  s'.pathParent = s.pathParent
70  |  |  if !openSet.Contains(s') then
71  |  |  |  openSet.Insert(s')
72  |  |  else
73  |  |  |  openSet.Update(s')
74  end
```

Fonte: Autor.

Todavia, caso o campo de visão ponderado esteja obstruído ou com peso elevado para indicar rota adiante na mesma direção, é realizado o chamado *"Path 1"* (Algoritmo 7) realizado pelo Theta*, que também utiliza o custo na decisão. Neste, o tratamento de altura é feito verificando o custo para cada vizinho imediato do nodo atual no processamento da busca.

Algoritmo 7: Algoritmo que realiza o *"Path 1"*, quando o campo de visão é obstruído.

```
75  doPathOne(s, s', maxAng)
76  |  ang = verifiesLargerNormalVector(s')
77  |  if ang >= maxAng then
78  |  |  continue;
79  |  cost = switch (s'.cellDirection)
79  |  newMovementCost = s.Gcost + c(s, s') + cost
80  |  if newMovementCost < s'.Gcost then
81  |  |  s'.Gcost = newCostToMove
83  |  |  s'.pathParent = s
84  |  |  if !openSet.Contains(s') then
85  |  |  |  openSet.Insert(s')
85  |  |  else
86  |  |  |  openSet.Update(s')
87  end
```

Fonte: Autor.

O Algoritmo 8 refere-se à decisão de qual nodo será atingido de acordo com a direção do movimento e o custo relativo a esta movimentação. O custo do movimento é calculado

para a normal referente à direção de saída do nodo ($normalFrom$) e referente à normal de entrada do nodo vizinho ($normalTo$) para o qual o movimento se direciona. É realizada uma verificação para saber se cada normal refere-se a um aclive. Caso afirmativo, o custo é computado e utilizado no algoritmo de busca.

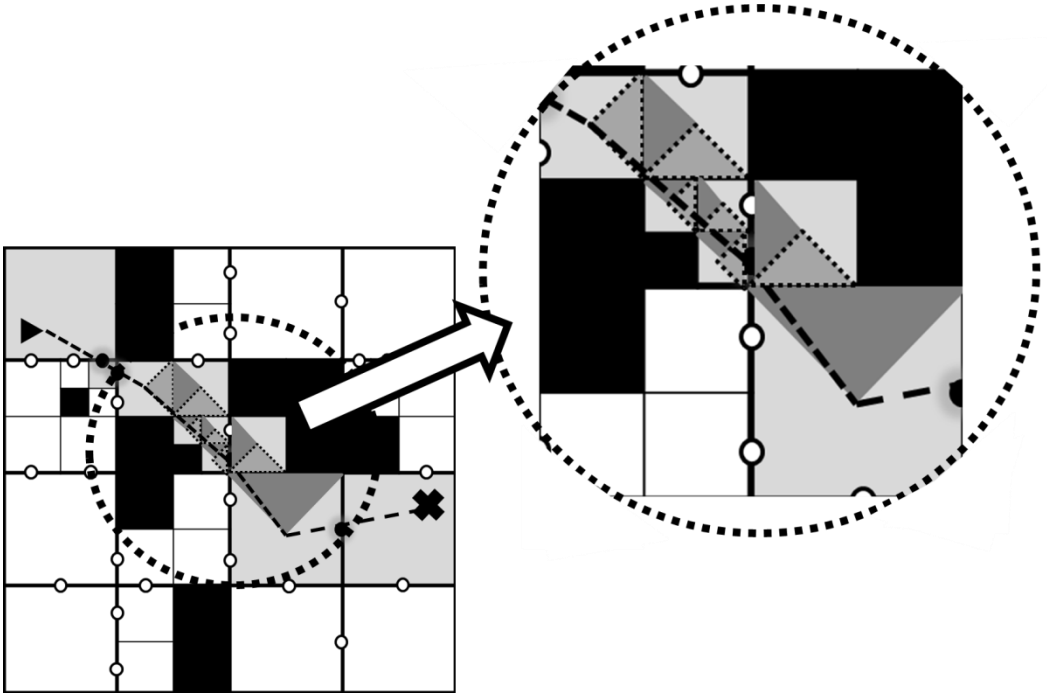
Algoritmo 8: Verificação da direção do nodo visitado e se é um aclive, assim o custo é computado.

```
88. switch(s'.cellDirection)
89.       case North:
90.           normalFrom = s.normalNorth
91.           normalTo = s'.normalSouth
92.           if isClimbing then
93.               cost = computeCost(normalFrom, normalTo)
94.       case West:
95.           normalFrom = s.normalWest
96.           normalTo = s'.normalEast
97.           if isClimbing then
98.               cost = computeCost(normalFrom, normalTo)
99.       case South:
100.          normalFrom = s.normalSouth
101.          normalTo = s'.normalNorth
102.          if isClimbing then
103.              cost = computeCost(normalFrom, normalTo)
104.      case East:
105.          normalFrom = s.normalEast
106.          normalTo = s'.normalWest
107.          if isClimbing then
108.              cost = computeCost(normalFrom, normalTo)
109.end
```

Fonte: Autor.

Para o campo de visão, a ideia de entrada e saída funciona da mesma forma. Todos os nodos pelos quais a linha do campo de visão passa têm seus custos de entrada e saída computados, como exibido na Figura 5.1. Na imagem, as regiões triangulares de saída estão representadas na cor cinza claro, com borda tracejada. As regiões triangulares de entrada estão representadas na cor cinza escuro. Os custos de entrada e saída de cada nodo são calculados e somados aos demais nodos que estão inclusos no campo de visão, formando o custo do campo de visão míope.

Figura 5.1.1: Nodos inclusos na linha do campo de visão em um trecho da rota encontrada pelo algoritmo, com suas regiões de entrada e saída demarcadas.



Fonte: Autor.

O Theta* proporciona maior qualidade em seus caminhos, contudo, como abordado na seção 2.6, exige maior tempo de execução e uso de memória. Quando abordado o tratamento de inclinações, o tempo se torna ainda maior. Visando resolver os problemas referentes a estas questões, foi incluída a abordagem hierárquica ao trabalho, diminuindo a complexidade da busca e resultando em menor tempo de execução desta, como comprovado em (Botea *et al.*, 2004). Ainda, vimos que o Theta* aliado ao HPA* resulta em uma associação de sucesso, visto que os benefícios de ambos os algoritmos são garantidos (Van Elswijk *et al.*, 2013).

Obtendo a versão do Theta* que trata as dificuldades em relação aos desníveis do terreno e aliando-o à estrutura hierárquica, foi desenvolvido o algoritmo objetivo deste trabalho: o HPATheta*, que utiliza o algoritmo Theta* com progressão na altura e técnica hierárquica.

Assim, a estrutura de representação do terreno, descrita por uma *quadtree*, é abstraída em clusters, que são representados por nodos em níveis mais altos da *quadtree*. O pré-processamento ocorre buscando caminhos internos em cada cluster, encontrados utilizando um algoritmo Theta* voltado ao tratamento de inclinação do terreno, de todos os nodos

trafegáveis de fronteira para os demais. Isso segue a ideia apresentada pelo HPA*, que foi descrita na seção 2.6. Ainda, são acoplados os nodos vizinhos de fronteira entre clusters, conectando-os dessa forma. Em tempo de execução, ao informar os pontos iniciais e finais, são encontrados os clusters de interesse e o algoritmo Theta* modificado para tratar inclinações é executado entre eles para fazer a busca o caminho concreto.

Com a implementação, além de utilizar benefícios específicos do Theta* e do HPA*, foi utilizado em conjunto com o tratamento de desníveis do terreno, o qual utiliza uma forma de campo de visão ainda não descrita na literatura. A implementação do algoritmo se concretizou com o processo de todas as pesquisas e avaliações previamente apresentados. Tendo resultados visualmente positivos, foi iniciada a fase de experimentos a fim de avaliar desempenho e qualidade.

6 EXPERIMENTOS E RESULTADOS

Os experimentos desenvolvidos neste trabalho visam avaliar a eficiência do algoritmo proposto no tratamento de inclinações ao buscar caminho entre dois pontos, de acordo com as restrições do agente envolvido no problema. A hipótese é que o tratamento de inclinações do terreno durante a busca de caminho forneça caminhos menos custosos. Isto significa traçar rotas que não ofereçam riscos à movimentação do agente, desviando de regiões com inclinações acentuadas.

Os experimentos para avaliar o algoritmo desenvolvido foram realizados utilizando dois grandes terrenos reais como mapas virtuais de navegação. Um dos terrenos, denominado R, possui como característica principal de seu relevo uma grande densidade de montanhas, o que permite testar o algoritmo que possui intuito de tratar inclinações no terreno. Por outro lado, o terreno denominado F apresenta poucos desníveis em seu relevo. Os dados referentes às características de cada terreno são apresentados na Tabela 1.

Para cada terreno, quatro algoritmos diferentes são executados, para avaliação de resultados e comparação entre os mesmos. Os algoritmos selecionados foram o Theta* em sua forma padrão, o Theta* modificado para tratar alturas, o algoritmo hierárquico que utiliza Theta* em sua forma padrão, chamado de HPT*, e o HPATheta*, algoritmo proposto. Para cada algoritmo, foram medidos e armazenados os valores referentes ao comprimento do caminho, tempo de execução do *pathfinding* e custo do caminho. O custo é a métrica de maior relevância uma vez que reflete se o algoritmo evitou inclinações que poderiam impactar na segurança da movimentação, contornando regiões de risco. Os algoritmos são testados para 30.000 pontos, iniciais e finais do mapa de navegação, gerados aleatoriamente.

Tabela 1 - Características dos terrenos virtuais e protocolo de testes

	F	**R**
Tamanho do terreno	30km x 56km	33km x 33km
Estrutura de representação do terreno	*Quadtree* hierárquica	*Quadtree* hierárquica
Percentual de nodos bloqueados	34%	24%
Percentual de inclinações relevantes	47%	59%
Número de nodos folha	115624	10753
Métricas de caminho	Custo do caminho resultante Tempo de execução da busca Comprimento do caminho HPATheta* HPT*	Custo do caminho resultante Tempo de execução da busca Comprimento do caminho HPATheta* HPT*
Algoritmos de busca	Theta* com tratamento de altura Theta* padrão	Theta* com tratamento de altura Theta* padrão
Conjunto de dados dos testes	30 mil pares de pontos iniciais e finais na representação do terreno virtual.	30 mil pares de pontos iniciais e finais na representação do terreno virtual.

Para analisar estatisticamente os resultados obtidos nos experimentos, foram utilizados modelos de regressão linear generalizada (McCullagh & Nelder, 1989). Esses modelos foram implementados usando o pacote de software estatístico R. Apesar da complexidade de tais modelos generalizados, eles apresentam, de forma confiável, medidas de tendência central e disseminação, além de serem mais robustas e mais frequentemente utilizadas do que modelos não paramétricos de regressão. Entre outras razões, as técnicas de regressão generalizadas permitem explorar diferentes tipos de distribuições na construção dos modelos para descrever os resultados experimentais. Particularmente devido ao tipo de resultados obtidos nos experimentos realizados, a distribuição Gama foi explorada na construção dos modelos de regressão. Com efeito, essa distribuição representa melhor os valores reais positivos, que são

os valores para o tempo de percurso e o custo do caminho resultante medidos nos experimentos. Na prática, os resultados de cada algoritmo testado HPATheta*, HPT*, Theta* com altura e Theta* sem altura foram incluídos nos modelos de regressão, permitindo comparar estatisticamente essas técnicas.

Os modelos de regressão para a comparação do tempo computacional e do custo nos caminhos resultantes dos algoritmos testados foram descritos como g (mu) = Beta_0 + Beta_1 * X + Beta2 + D1 + Beta3 + D2 + Beta4 + D3, onde g é a função de log. Na prática, este modelo representa valores de tempo de percurso em função das distâncias resultantes entre as posições de início e de objetivo no terreno virtual, onde tais valores de distância são representados pela variável X do modelo. Da mesma forma, esse tipo de representação também foi usado para descrever os resultados de custo do caminho resultante em função das distâncias resultantes entre os pontos iniciais e finais. Nesta, as variáveis *dummy* D1 - D3 foram utilizadas para representar as técnicas comparadas, onde todas as comparações foram feitas em relação a um método base, o proposto neste trabalho (HPATheta*). Assim, os modelos de regressão descrevem o tempo computacional do algoritmo HPATheta* quando D1 = D2 = D3 = 0. Respectivamente, eles descrevem o algoritmo HPT* quando D1 = 1, o algoritmo Theta* com altura quando D2 = 1 e o algoritmo Theta* sem altura quando D3 = 1. Para obter resultados para cada um desses algoritmos, a variável Di que representa o i-ésima algoritmo precisa ter o valor 1, enquanto as outras variáveis D precisam ter o valor = 0.

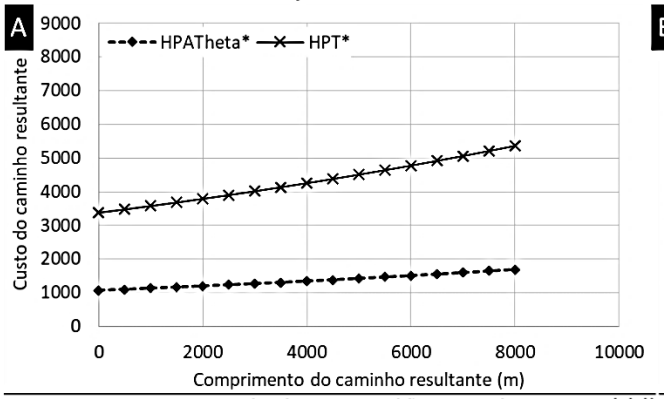
Na comparação dos algoritmos HPATheta*, HPT*, Theta* com altura e Theta* sem altura, as estimativas resultantes representam os valores médios em relação ao tempo computacional e o custo do caminho resultante em relação a diferentes valores de distância. Devido ao fato de que os resultados de tais funções de regressão construídas podem ser qualquer valor real, a função de vinculação usada nos modelos de regressão é a função de log. Para desenvolver a análise estatística, foi definido um nível de significância de 0,01 (1%). Assim, as hipóteses H0 e H1 foram definidas com Beta_i = 0 ou Beta_i ≠ 0, respectivamente, para i = 1, 2, 3 e 4. Então, o alfa foi comparado com o valor-p. Se alfa> P-valor então H0 não é rejeitado; caso contrário, H0 é rejeitado. Beta_1> 0 significa que distâncias maiores entre as posições de origem e de destino implicam em tempos mais longos para encontrar caminhos em tais distâncias. Para Beta_i, i = 2, 3 e 4, Beta_i = 0 significa que o método Di é equivalente ao método base. Beta_i > 0 significa que o Di é mais lento que o método base. Beta_i < 0 significa que Di é mais rápido que o método base.

Nos resultados obtidos para todas as técnicas testadas, conforme apresentado nas Figuras 6.2.1 e 6.2.2, os valores de P obtidos com os modelos de regressão são todos próximos de zero e inferiores ao nível alfa considerado de 0,01; portanto, é possível afirmar que todos os resultados são estatisticamente significativos. Além disso, os modelos de regressão mostram que existem diferenças significativas entre as técnicas comparadas. Na prática, todos os valores resultantes relativos ao tempo de computacional e ao custo resultante para as distâncias entre as posições de início e de meta em todas as técnicas de HPT*, Theta* com altura e Theta* sem altura são diferentes da técnica HPATheta* da linha de base.

Como todas as estimativas de custo médio na Figura 6.2.1 são valores positivos, os modelos de regressão mostram que os algoritmos de *pathfinding* resultam em maior custo para realizar caminhos computados com o uso dos algoritmos HPT*, Theta* com altura e Theta* sem altura do que com o uso do algoritmo HPATheta*.

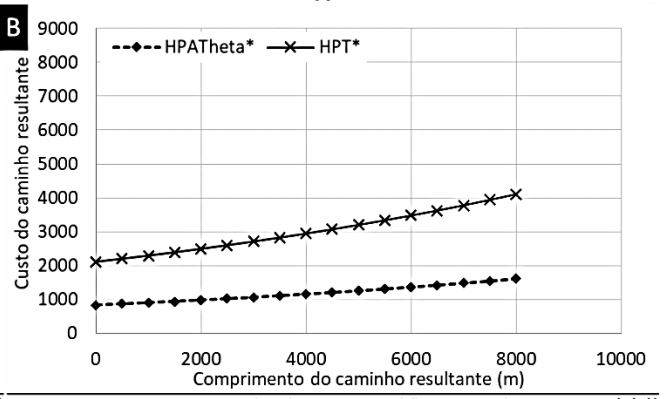
Figura 6.2.1: Resultados para Custo em relação à Distância dos testes realizados.

F

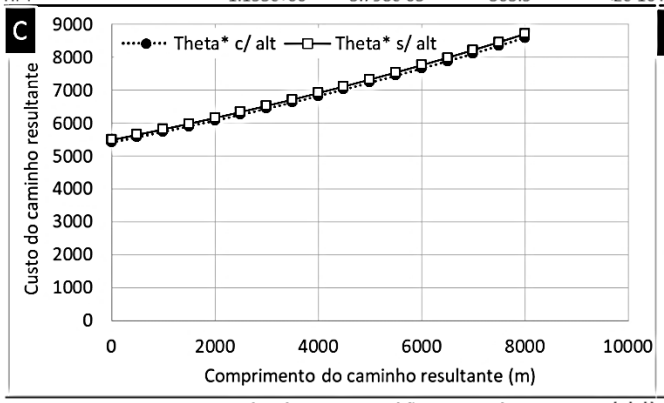


	Estimativa	Erro padrão	Valor t	Pr(>|t|)
Intercepto	6.973e+00	4.458e-03	1564.1	<2e-16
HPATheta*	5.773e-05	2.192e-07	263.3	<2e-16
HPT*	1.153e+00	3.798e-03	303.5	<2e-16

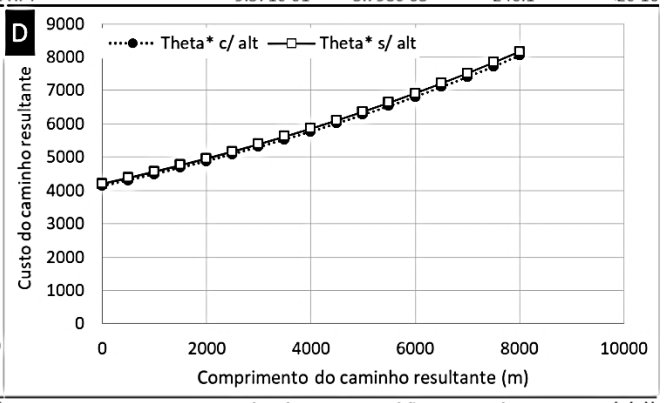
R



	Estimativa	Erro padrão	Valor t	Pr(>|t|)
Intercepto	6.720e+00	4.357e-03	1542.1	<2e-16
HPATheta*	8.306e-05	2.672e-07	310.9	<2e-16
HPT*	9.371e-01	3.798e-03	246.1	<2e-16



	Estimativa	Erro padrão	Valor t	Pr(>|t|)
Intercepto	6.973e+00	4.458e-03	1564.1	<2e-16
Theta* c/ altura	1.625e+00	3.852e-03	421.9	<2e-16
Theta* s/ altura	1.638e+00	3.853e-03	425.2	<2e-16

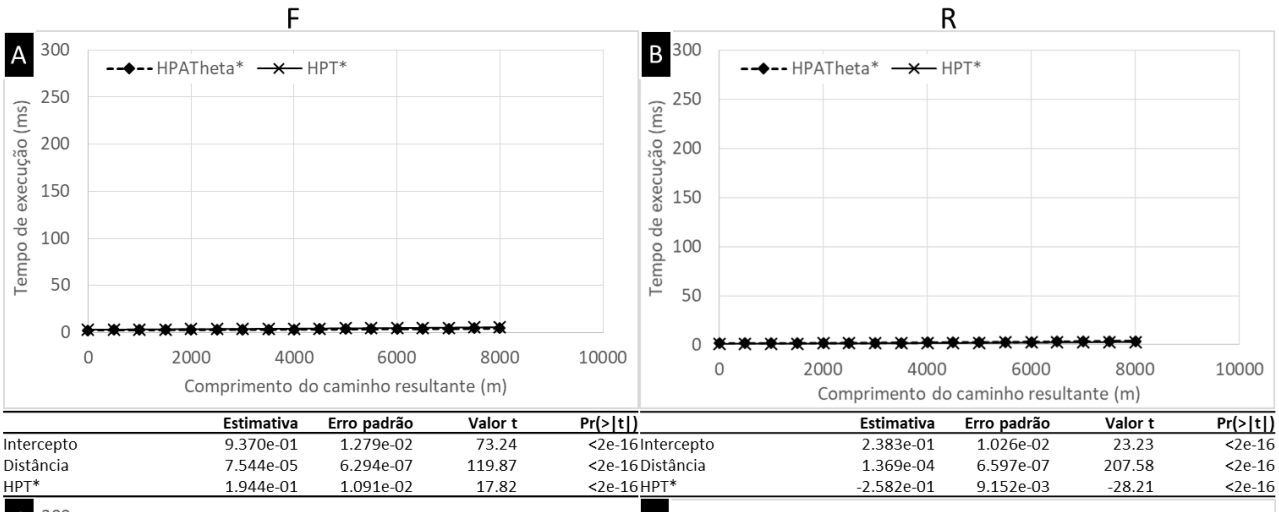


	Estimativa	Erro padrão	Valor t	Pr(>|t|)
Intercepto	6.720e+00	4.357e-03	1542.1	<2e-16
Theta* c/ altura	1.609e+00	3.827e-03	420.5	<2e-16
Theta* s/ altura	1.624e+00	3.829e-03	424.1	<2e-16

Fonte: Autor.

Quando se trata do tempo de execução, os dados do modelo informam que apenas o algoritmo HPT* retorna caminhos em tempos computacionais menores em relação ao algoritmo base, como mostrado na Figura 6.2.2 (visto que a estimativa apresenta valor negativo para custo médio). Este é um resultado coerente, visto que o HPATheta* tende a explorar mais nodos durante a busca pelo menor custo em alguns casos, enquanto o HPT* busca apenas o caminho mais curto. Apesar disso, os tempos de busca destes dois algoritmos são bastante aproximados. Ainda, os resultados demonstram diferenças acentuadas entre os tempos de execução dos algoritmos hierárquicos em relação aos não hierárquicos. Isto ocorre devido à quantidade nodos abertos pelos algoritmos, visto que ao utilizar busca hierárquica o número de nodos abertos é limitado pelos clusters, refletindo portanto, no tempo computacional.

Figura 6.2.2: Resultados para Tempo em relação à Distância dos testes realizados.



	Estimativa	Erro padrão	Valor t	Pr(>|t|)
Intercepto	9.370e-01	1.279e-02	73.24	<2e-16
Distância	7.544e-05	6.294e-07	119.87	<2e-16
HPT*	1.944e-01	1.091e-02	17.82	<2e-16

	Estimativa	Erro padrão	Valor t	Pr(>|t|)
Intercepto	2.383e-01	1.026e-02	23.23	<2e-16
Distância	1.369e-04	6.597e-07	207.58	<2e-16
HPT*	-2.582e-01	9.152e-03	-28.21	<2e-16

	Estimativa	Erro padrão	Valor t	Pr(>|t|)
Intercepto	9.370e-01	1.279e-02	73.24	<2e-16
Theta* c/ altura	2.960e+00	1.090e-02	271.51	<2e-16
Theta* s/ altura	4.135e+00	1.113e-02	371.41	<2e-16

	Estimativa	Erro padrão	Valor t	Pr(>|t|)
Intercepto	2.383e-01	1.026e-02	23.23	<2e-16
Theta* c/ altura	2.857e+00	9.067e-03	315.13	<2e-16
Theta* s/ altura	2.409e+00	9.062e-03	265.84	<2e-16

Fonte: Autor.

7 CONCLUSÃO

Algoritmos de *pathfinding* são algoritmos que buscam rotas trafegáveis para agentes baseados em algum critério de prioridade, em geral, o caminho mais curto. Contudo, ter um algoritmo que considere as restrições dos agentes envolvidos no problema, torna-se relevante à medida que a complexidade do terreno virtual aumenta. Em sistemas de simulação, que buscam realizar simulações o mais próximo possível da realidade, tratar adequadamente estas restrições e evitar falhas torna-se um importante objetivo.

Visto que o objetivo deste trabalho é propor um algoritmo de busca de caminhos que trate desníveis em um terreno virtual, é possível concluir que o algoritmo desenvolvido cumpriu sua função com êxito. O tratamento do problema de inclinações no terreno reflete-se no custo obtido a partir do caminho encontrado. Uma vez que os resultados obtidos demonstram que o algoritmo HPATheta* obtém menor custo para mesmos intervalos de distância, em uma mesma região, conclui-se que o trabalho atingiu os objetivos propostos, sendo, portanto, bem-sucedido. Assim, o algoritmo implementado proporciona:

1) Caminhos seguros, respeitando às restrições parametrizadas para cada agente;
2) Caminhos de maior qualidade, visto que o algoritmo trata custo do caminho ao passo em que realiza suavização do mesmo;
3) Tempo de execução otimizado, uma vez que associa a técnica de busca hierárquica aos benefícios mencionados em 1) e 2).

Tratando alturas em terrenos, os caminhos resultantes de um algoritmo de busca tornam-se mais confiáveis, de forma a evitar falhas na navegação durante a execução do sistema no qual o algoritmo é utilizado, satisfazendo, assim, o objetivo deste trabalho.

8 REFERÊNCIAS BIBLIOGRÁFICAS

ANGUELOV, B. **Video game pathfinding and improvements to discrete search on grid-based maps.** 2011. (PhD Diss.). Faculty of Engineering, Built Environment and Information Technology, University of Pretoria, Pretoria, South Africa.

ALGFOOR, Z. A.; SUNAR, M. S.; KOLIVAND, H. A comprehensive study on pathfinding techniques for robotics and video games. **International Journal of Computer Games Technology,** v. 2015, p. 7, 2015. ISSN 1687-7047.

BLYTH, S. **Mountain watch: environmental change & sustainable developmental in mountains**. UNEP/Earthprint, 2002. ISBN 1899628207.

BOTEA, A. et al. **Pathfinding in games**: Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik 2013.

BOTEA, A.; MÜLLER, M.; SCHAEFFER, J. Near optimal hierarchical path-finding. **Journal of game development,** v. 1, n. 1, p. 7-28, 2004.

BRONDANI, J. R.; DE FREITAS, E. P.; SILVA, L. A. A task-oriented and parameterized (semi) autonomous navigation framework for the development of simulation systems. **Procedia Computer Science,** v. 112, p. 534-543, 2017. ISSN 1877-0509.

BRONDANI, J. R. et al. **Semi-Autonomous Navigation for Virtual Tactical Simulations in the Military Domain**. 8th International Conference on Simulation and Modeling (SIMULTECH). Porto, Portugal**:** 443-450 p. 2018.

CHEN, S.; SHI, G.; LIU, Y. **Fast path searching in real time 3D game**. 2009 WRI Global Congress on Intelligent Systems: IEEE. 3**:** 189-194 p. 2009.

DOOMS, A. Parallel multi-agent path planning in dynamic environments for real-time applications. 2013.

GANGANATH, N.; CHENG, C.-T.; CHI, K. T. A constraint-aware heuristic path planner for finding energy-efficient paths on uneven terrains. **IEEE transactions on industrial informatics,** v. 11, n. 3, p. 601-611, 2015. ISSN 1551-3203.

KAPADIA, M.; BADLER, N. I. Navigation and steering for autonomous virtual humans. **Wiley Interdisciplinary Reviews: Cognitive Science,** v. 4, n. 3, p. 263-272, 2013. ISSN 1939-5078.

NASH, A. et al. Theta^*: Any-angle path planning on grids. AAAI, 2007. p.1177-1183.

NILSSON, N. J.; NILSSON, N. J. **Artificial intelligence: a new synthesis**. Morgan Kaufmann, 1998. ISBN 1558604677.

PELECHANO, N.; FUENTES, C. Hierarchical path-finding for Navigation Meshes (HNA∗). **Computers & Graphics,** v. 59, p. 68-78, 2016. ISSN 0097-8493.

PÜTZ, S. et al. 3d navigation mesh generation for path planning in uneven terrain. **IFAC-PapersOnLine,** v. 49, n. 15, p. 212-217, 2016. ISSN 2405-8963.

RUSSELL, S. J.; NORVIG, P. **Artificial intelligence: a modern approach**. Malaysia; Pearson Education Limited, 2016.

SAMET, H. The quadtree and related hierarchical data structures. **ACM Computing Surveys (CSUR),** v. 16, n. 2, p. 187-260, 1984. ISSN 0360-0300.

SOUISSI, O. et al. Path planning: A 2013 survey. Proceedings of 2013 International Conference on Industrial Engineering and Systems Management (IESM), 2013, IEEE. p.1-8.

THORPE, C.; MATTHIES, L. Path relaxation: Path planning for a mobile robot. OCEANS 1984, 1984, IEEE. p.576-581.

VAN ELSWIJK, L.; SPRINKHUIZEN-KUYPER, I.; WIEDIJK, F. **Hierarchical Path-Finding Theta**. 2013.