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

Ricardo Kunde Schlsener

COMPUTAÇÃO DE CAMINHOS ENGANOSOS CONSIDERANDO CARACTERÍSTICAS TOPOGRÁFICAS DO TERRENO E REDES NEURAIS PROFUNDAS

Santa Maria, RS
2022

Ricardo Kunde Schlsener

COMPUTAÇÃO DE CAMINHOS ENGANOSOS CONSIDERANDO CARACTERÍSTICAS TOPOGRÁFICAS DO TERRENO E REDES NEURAIS PROFUNDAS

Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em Ciência da Computação, Área de Concentração em ciências exatas e da terra, da Universidade Federal de Santa Maria (UFSM, RS), como requisito parcial para obtenção do grau de **Bacharel em Ciência da Computação**.

ORIENTADOR: Prof. Luis Alvaro de Lima Silva

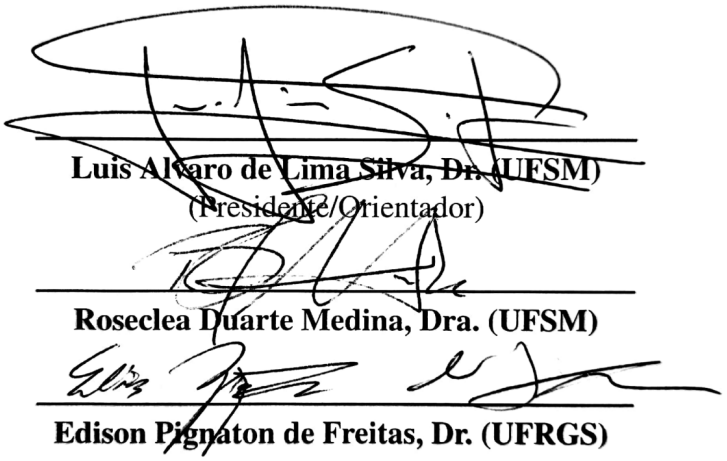
515
Santa Maria, RS
2022

Ricardo Kunde Schlsener

COMPUTAÇÃO DE CAMINHOS ENGANOSOS CONSIDERANDO CARACTERÍSTICAS TOPOGRÁFICAS DO TERRENO E REDES NEURAIS PROFUNDAS

Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em Ciência da Computação, Área de Concentração em ciências exatas e da terra, da Universidade Federal de Santa Maria (UFSM, RS), como requisito parcial para obtenção do grau de **Bacharel em Ciência da Computação**.

Aprovado em 10 de agosto de 2022:



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

Roseclea Duarte Medina, Dra. (UFSM)

Edison Pignaton de Freitas, Dr. (UFRGS)

Santa Maria, RS
2022

AGRADECIMENTOS

Quero primeiramente agradecer a minha família por estar sempre aqui por mim durante toda a minha vida.

Agradeço o meu orientador Luis Alvaro, junto com a banca Roseclea e Edison, por me guiar a conclusão deste trabalho e assim me ajudar a finalizar esta jornada acadêmica.

Também quero agradecer o Cristian e Gabriel por terem me ajudado diretamente com o desenvolvimento deste trabalho.

E por último quero agradecer todos os meus amigos por alegrar os meus dias e fazer tudo isso valer a pena.

RESUMO

COMPUTAÇÃO DE CAMINHOS ENGANOSOS CONSIDERANDO CARACTERÍSTICAS TOPOGRÁFICAS DO TERRENO E REDES NEURAIS PROFUNDAS

AUTOR: Ricardo Kunde Schlsener
ORIENTADOR: Luis Alvaro de Lima Silva

Existem diversas aplicações reais para algoritmos com características enganosas, e é uma das áreas da Inteligência Artificial que ainda tem muito a ser explorada, principalmente no contexto de busca de caminho (*pathfinding*). Quase todos os estudos sobre caminhos enganosos na literatura são aplicados em cenários bidimensionais, e existe poucos estudos sobre a otimização do processamento destes caminhos. O objetivo deste trabalho é investigar algoritmos voltados a computação de caminhos enganosos e aplica-los em terrenos que possuem características de relevo, analisando como estes caminhos são afetados pela consideração explícita de características topográficas do terreno. Este trabalho também explora a aplicação de redes neurais profundas para apoiar a computação destes caminhos. A partir disso é feita uma análise dos resultados para determinar o impacto que as técnicas analisadas apresentam nas características dos caminhos retornados e no seus tempos de processamento. Os resultados encontrados demonstram que os caminhos enganosos são efetivos em terrenos com relevo e também apresentam o grande potencial que as redes neurais profundas apresentam como forma de otimização da computação destes caminhos.

Palavras-chave: Engano. Busca de Caminhos. Redes Neurais Profundas.

ABSTRACT

COMPUTATION OF DECEPTIVE PATHS CONSIDERING THE TERRAIN'S TOPOGRAPHIC CHARACTERISTICS AND DEEP NEURAL NETWORKS

AUTHOR: Ricardo Kunde Schlsener
ADVISOR: Luis Alvaro de Lima Silva

There's a variety of real life applications for algorithms with deceptive characteristics, and it's an area in Artificial Inteligence with a lot still left to be explored, specially in the context of pathfinding algorithms. Most studies in the literature about deceptive pathfinding are applied in two-dimensional spaces, and there's little to no studies about the application of machine learning in this context. This work aims to investigate deceptive pathfinding algorithms and apply them in terrains with topographic characteristics, analysing the impact of explicitly considering these characteristics. This work also seeks to explore the application of deep neural networks as a way to optimize the computation of these paths. Afterwards, a statistical analysis of the results is made to determine the impact that these techniques have in the final path and in their processing time.

Keywords: Deep Neural Networks. Artificial Intelligence. Machine Learning. Deception. Pathfinding.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

ANN	*Artifical Neural Network*
DNN	*Deep Neural Network*
LDP	*Last Deceptive Point*
MDE	Modelo Digital de Elevação
TIF	*Tag Image File Format*
ReLU	*Rectified Linear Unit*
RMP	*Radius of Maximum Probability*
GLM	*Generalized Linear Model*

SUMÁRIO

1 INTRODUÇÃO

Busca de caminhos (BOTEA et al., 2013) (ALGFOOR; SUNAR; KOLIVAND, 2015) é uma área da Inteligência Artificial (IA) dedicada à pesquisa e aplicação de algoritmos de planejamento de caminhos em grafos, *grids* e outros tipos de cenários do mundo real. Algoritmos que tentam encontrar o caminho de menor distância são os mais comumente apresentados na literatura, com inúmeras técnicas já desenvolvidas e testadas em uma variedade de contextos. Porém, existem outros importantes focos de aplicação que estes algoritmos podem explorar que não buscam simplesmente encontrar o caminho mais curto, como por exemplo a computação de caminhos enganosos (MASTERS; SEBASTIAN, 2017). A computação de caminhos enganosos envolve planejar um caminho aonde um observador, podendo este ser uma pessoa ou um algoritmo de detecção avaliando o trajeto de um agente em movimento por um caminho, tem dificuldade de determinar as verdadeiras intenções deste agente, dado que o agente se desloca pelo caminho de uma posição inicial até uma posição alvo.

Algoritmos de busca de caminhos enganosos têm aplicação ampla, podendo ser usados na implementação de personagens em jogos digitais para criar comportamentos mais realistas e engajantes para os jogadores, no planejamento e simulações de caminhos estratégicos (MACAL, 2016b), em veículos terrestres não-tripulados e outros tipos de aplicações envolvendo robôs móveis para capacitar esses agentes a despistar sistemas de observação em situações do mundo real. Em geral, a construção e teste de algoritmos enganosos podem avançar a área de IA no sentido de melhor modelar importantes comportamentos humanos nos algoritmos inteligentes construídos.

Embora existam trabalhos na literatura sobre busca de caminhos enganosos (MASTERS; SEBASTIAN, 2017) (CAI et al., 2020), essa área de pesquisa ainda é relativamente inexplorada. Muitos dos algoritmos existentes ainda não foram avaliados e testados em profundidade em uma variedade de contextos, incluindo terrenos com relevo, que são uma das formas mais comuns de aplicação de algoritmos de busca de caminhos, tanto em jogos digitais quanto em simulações.

Esses algoritmos também podem apresentar elevados tempo de processamento, e mesmo que existam pesquisas para a otimizar essas técnicas enganosas (XU et al., 2020a), ainda há várias importantes técnicas de otimização como redes neurais profundas (Deep Neural Networks - DNN) (ALOM et al., 2019) que ainda podem ser exploradas nesta área. Em jogos digitais ou em simulações a tempo real, podem ser calculados múltiplos caminhos ao mesmo tempo, e estes cálculos acontecem enquanto outras partes do programa estão sendo executadas, então é necessário que a computação dos caminhos dos agentes seja próximo de instantâneo, caso o contrário estes agentes vão agir de forma anormal, ficando parados sem fazer nada por um tempo estendido, ou até pior, o programa todo vai ficar travado enquanto se espera o cálculo destes caminhos.

O objetivo deste trabalho é investigar algoritmos voltados a computação de caminhos

enganosos e aplica-los em terrenos que possuem características de relevo, analisando como estes caminhos são afetados pela consideração explícita de características topográficas do terreno. A aplicação analisada nesta pesquisa considera um ambiente com múltiplos possíveis objetivos, onde apenas um deles é o alvo verdadeiro que o agente deve atingir no fim do seu caminho. Essa abordagem é baseada no cálculo de um trajeto onde um observador analisando os vários passos do caminho computado tenha dificuldade de determinar qual dos possíveis objetivos é o alvo deste agente. O trabalho também trata o problema de tempos de processamento elevados associados a computação de caminhos enganosos com a aplicação de aprendizado de máquina, propondo, implementando e testando o emprego de DNN como forma de otimizar as computações realizadas.

Para realizar o estudo das técnicas aplicadas, a análise de diferentes versões dos algoritmos implementados é realizada de acordo com critérios como: i) tempo de processamento, ii) número de nodos do terreno analisados na determinação do caminho, iii) custo/distância do caminho resultante, iv) quão enganoso os caminhos gerados são, e v) quais são as principais mudanças no caminho planejado devidas a topografia do terreno. Os resultados dos testes com os diferentes algoritmos implementados são analisados de acordo com modelos estatísticos de regressão linear (MCCULLAGH; NELDER, 1983), permitindo realizar comparações entre execuções dos algoritmos para situações de teste planejadas. Além de permitir comparar os algoritmos implementados de acordo com diferentes métricas, o objetivo é verificar se os caminhos computados nestes terrenos com relevo mantém as características enganosas originalmente propostas para os algoritmos testados.

Em particular, o trabalho desenvolve uma pesquisa que adapta e expande para um novo contexto as técnicas de engano principalmente apresentadas por Masters e Sebastian (2017) e a aplicação de DNN em terrenos com relevo apresentadas por Neisse et al. (2022). O trabalho também demonstra como combinar essas técnicas com o intuito de aprofundar o entendimento de busca de caminhos enganosos e de engano na construção de sistemas de simulação baseados em agentes (MACAL, 2016a) para diferentes problemas de aplicação.

2 REFERENCIAL TEÓRICO

Este capítulo apresenta conceitos necessários para o desenvolvimento e entendimento deste trabalho. O capítulo apresenta uma base teórica sobre busca de caminhos, com um foco no algoritmo A*, e técnicas que permitem a consideração de caminhos em terrenos com características topográficas. O trabalho também apresenta um estudo sobre engano, analisando esse assunto no contexto de busca de caminhos pesquisada neste trabalho. Por último, trabalhos relacionados envolvendo busca de caminhos enganosos, aplicações de DNN em busca de caminhos, e busca de caminhos em terrenos com relevo são revisados e analisados.

2.1 BUSCA DE CAMINHOS

A busca de caminhos (ALGFOOR; SUNAR; KOLIVAND, 2015) tem como objetivo o desenvolvimento de técnicas e algoritmos para calcular a rota para um agente se movimentar de um ponto inicial até um ponto destino em um terreno considerado. Os algoritmos mais estudados são aqueles que buscam um caminho ótimo, ou seja, um caminho de menor custo possível. Além de existir caminhos semi-ótimos, que são importantes para várias aplicações, e que podem ser computados de forma otimizada em termos de tempo de processamento e uso de memória, também existem outros comportamentos que podem ser desejados na determinação de um caminho. Por exemplo, caminhos que permitam otimizar o consumo de energia e as restrições de movimentação dos agentes (CHOI et al., 2012) ou caminhos que sejam enganosos (MASTERS; SEBASTIAN, 2017), ou ambos, o que é o foco deste trabalho.

De forma geral, algoritmos de busca de caminho são normalmente computados com o emprego de grafos, onde passos necessários para determinação do caminho resultante são realizados nodo a nodo. Estes grafos podem ser representados em cenários digitais de diferentes formas (ALGFOOR; SUNAR; KOLIVAND, 2015), com *grids* de diferentes formatos e ligações, ou até com o emprego de *meshes* em cenários 3D. Embora representações mais otimizadas de terrenos virtuais possam ser investigadas, permitindo melhor capturar as características do terreno na estrutura de dados usada, este trabalho utiliza uma das representações mais populares: um *grid* retangular de 2 dimensões que pode ser salvo como uma matriz em memória.

Um domínio onde a busca de caminho é explorada é uma tupla $D = \langle N, E, e \rangle$, onde:

- N é um conjunto de nodos;
- $E \subseteq NxN$ é um conjunto de arestas entre os nodos; e
- $c : E \rightarrow \mathbb{R}$ retorna o custo para atravessar entre os nodos. Neste trabalho, é importante observar que terrenos com topografia apresentam valores de custo c que são associados a inclinação do terreno entre os nodos considerados.

Um caminho π que atravessa um domínio onde a busca de caminho é realizada é uma sequência de nodos $\pi = n_0, n_1, ..., n_k$ de forma que $(n_i, n_{i+1}) \in E$ para cada i $\in$ 0, 1, ..., k-1. É utilizado π^i para representar o (a posição) passo i de um caminho π, e $|\pi|$ para representar a distância de π, ou seja, o número de arestas no caminho π. O custo de um caminho π é dado pela soma dos custos das arestas atravessadas pelo caminho computado, $cost(\pi) = \sum_{i=0}^{k-1} c(\pi^i, \pi^{i+1})$.

Um problema de busca de caminho é uma tupla $\langle D, s, g \rangle$, onde:

- $D = \langle N, E, c \rangle$ é um domínio onde a busca de caminho é computada;
- $s \in N$ é a posição de início; e
- $g \in N$ é a posição objetivo.

A solução de um problema de busca de caminho é um caminho π do domínio D em que $\pi^0 = s$ e $\pi^{|\pi|} = g$. Uma solução ótima é uma solução com o menor custo entre todas as possíveis soluções, representado por $optc(s, g)$.

Quando a efetividade de um algoritmo de busca de caminho é analisada existem diversos critérios a serem considerados. Esses critérios podem se referir ao caminho computado ou a execução do algoritmo. Para o caminho resultante, é possível analisar o custo do caminho, a suavidade do caminho (caminhos com menor número de zig-zags devidos a várias razões), e também parâmetros mais específicos como gasto de energia ou nível de engano do caminho. Em termos da execução do algoritmo, é possível analisar o tempo de execução, número de nodos abertos, uso de memória, entre outros (STURTEVANT, 2012). Tal como explorado neste trabalho, o estudo destes critérios a partir de caminhos obtidos em um conjunto de testes realizados pode ser feito de forma estatística, utilizando resultados de execuções dos algoritmos com diferentes parâmetros em diferentes cenários.

2.2 ALGORITMO A*

O algoritmo A* (HART; NILSSON; RAPHAEL, 1968) é um algoritmo fundamental da área de busca de caminhos, sendo usado e otimizado para inúmeras áreas de aplicação apresentadas na literatura (BOTEA et al., 2013). O A* é um algoritmo de busca heurístico que tem a Equação 2.1 como Equação de custo, onde *g(n)* é o custo até o nodo n e *f(n)* é uma função heurística. As funções heurísticas mais comumente usadas são a distância Euclidiana ou de Manhattan do nodo n até o destino.

$$f(n) = g(n) + h(n) \tag{2.1}$$

O algoritmo trabalha com uma lista de nodos abertos, começando apenas com o nodo inicial. Então ele entra em um *loop* onde é selecionado o nodo de menor custo entre os abertos. O algoritmo então verifica se o nodo atual é o objetivo, e calcula o custo dos nodos vizinhos ao

nodo atual. Os vizinhos são adicionados a lista de nodos abertos, mantendo uma ligação entre o nodo vizinho e o atual para que o caminho completo possa ser retornado no final do processo de busca. Isto se repete até o objetivo ser encontrado ou a lista de nodos abertos ficar vazia, o que significa que não existe um caminho entre o nodo inicial e o nodo objetivo.

Neste trabalho, o algoritmo A* é aplicado em um cenário com variação de altura, assim o custo dos nodos vizinhos leva em consideração a variação de altura entre eles. O pseudocódigo do A* é apresentado no Algoritmo 1. O uso da função heurística faz com que o algoritmo priorize melhor os nodos do mapa a serem analisados, resultando em um número reduzido de nodos abertos, consequentemente, o tempo de execução do algoritmo é menor.

Algorithm 1 A*

```
1:  função ASTAR(grafo, inicio, objetivo)
2:      abertos.inserir(inicio)
3:      fechados ← novaFila()
4:      enquanto !abertos.vazio() faça
5:          atual ← abertos.removeFrente()
6:          fechados.inseir(atual)
7:          se atual == objetivo então
8:              retorna atual
9:          fim se
10:         Para cada vizinho ∈ atual.vizinhos faça
11:             se !fechados.contem(vizinho) então
12:                 G ← atual.g + custoPasso(atual, vizinho)
13:                 se G < vizinho ou !abertos.contem(vizinho) então
14:                     vizinho.anterior ← atual
15:                     vizinho.g ← G
16:                     vizinho.h ← heurística(vizinho, objetivo)
17:                     se vizinho ∉ abertos então
18:                         abertos.inserir(vizinho)
19:                     fim se
20:                 fim se
21:             fim se
22:         fim para
23:     fim enquanto
24:     retorna nenhum caminho
25: fim função
```

Mapas com características topográficas podem ser armazenados como imagens ou matrizes que representam a altura do relevo do terreno em cada ponto do mapa. Para os fins de cálculos de busca de caminho, eles podem ser interpretados ou diretamente representados como grafos ponderados onde o custo ou peso das arestas é determinado pela variação de relevo, ou inclinação do relevo, entre posições/nodos consecutivos do mapa. Porém, há várias considerações a serem feitas quando existe a necessidade de interpretar as informações de elevação do terreno. Primeiro, existe a questão de qual tipo de agente terrestre vai usar os resultados do algoritmo de busca de caminhos. Considerando um robô terrestre ou até uma pessoa, existe certas

inclinações que são ingrimes demais para a movimentação. Tais regiões do terreno então podem ser consideradas como áreas que devem ser evitadas, ou mesmo obstáculos intransponíveis.

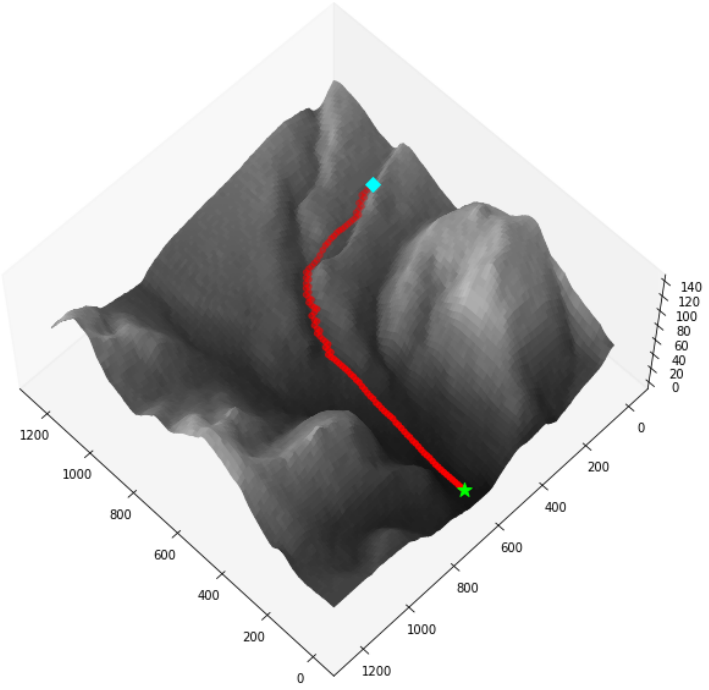
Também existe a consideração de quais critérios que estão sendo otimizados como parte do planejamento dos caminhos, já que a presença das características topográficas podem impactar as computações sendo consideradas. Em um terreno com relevo, por exemplo, a otimização de gasto de energia (CHOI et al., 2012) do agente quando em movimento pelo caminho computado ganha mais relevância muitas vezes em detrimento da menor distância a ser percorrida.

Neste trabalho, os caminhos são otimizados para engano e distância, e não é considerado nenhuma limitação de movimentação associada as inclinações encontradas no relevo do terreno. O algoritmo A* é utilizado em diversos passos da implementação para se determinar o caminho e o custo entre duas posições do mapa. Para que o A* possa ser aplicado em terrenos com altura e relevo, é necessário utilizar a distância Euclidiana tridimensional (Equação 2.2) para determinar o custo do passo de movimentação local no terreno e o valor da heurística na Equação 2.1.

$$dist = \sqrt{(p_x - q_x)^2 + (p_y - q_y)^2 + (p_z - q_z)^2} \tag{2.2}$$

A Figura 2.1 ilustra um caminho resultante do algoritmo A* para terrenos com altura e relevo.

Figura 2.1 – Exemplo de caminho computado pelo algoritmo A* ajustado para considerar as características de relevo de um terreno.



Fonte: Próprio autor

2.3 ENGANO

Para a IA, engano se refere a algoritmos e técnicas que tentam mascarar certas informações de observadores, como um objetivo a ser alcançado por um agente, por exemplo. Estes observadores podem ser humanos ou algoritmos automatizados voltados para o reconhecimento de objetivos. Para determinar se uma estratégia é enganosa ou não, primeiro é necessário determinar um ponto de vista a ser enganado, ou seja, um modelo de identificação que deve ser contrariado pelo engano.

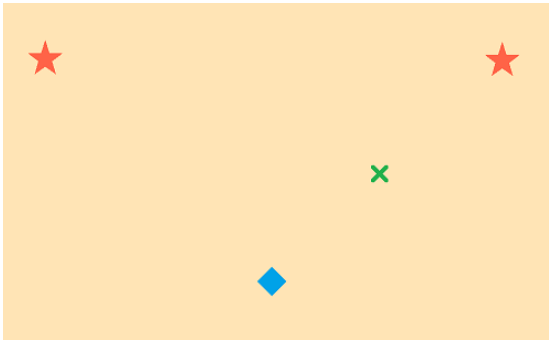
Dependendo do contexto em que este engano está sendo aplicado, o que constituiu um comportamento enganoso varia, já que os objetivos variam (HURWITZ; MARWALA, 2007), (FRAUNHOLZ et al., 2018). Por exemplo, em um jogo entre dois oponentes, onde o jogador 1 tem apenas uma ideia do que o jogador 2 deve fazer para ganhar, e vai tentar adivinhar o objetivo deste oponente para interceptar esta estratégia, se o jogador 2 jogar de uma forma que pareça que ele têm um objetivo diferente do seu verdadeiro objetivo, ou seja, agindo contra os seus próprios interesses, o jogador 1 vai se preparar para o objetivo errado, ou não saber para qual objetivo se preparar (ter dificuldade em predizer o objetivo do adversário). Desta forma, o modo de atuação do jogador 1 vai ser ineficiente quando o jogador 2 estiver realmente indo em frente para o seu objetivo verdadeiro. Neste caso, o jogador 2 tinha que conhecer as expectativas do jogador 1 para subverte-las. Computacionalmente, essas expectativas tomam forma de modelos de reconhecimento, contendo certas regras para determinar a intenção de um agente.

Este trabalho investiga a aplicação de engano na busca de caminhos em terrenos com características topográficas. Existem diferentes formas de analisar engano no contexto de busca de caminhos (Root, Mot e Feron (2005), por exemplo). Porém, este trabalho é baseado no modelo apresentado por Masters e Sardina (2017), que propõe um método de identificação de objetivos de um agente para ambientes onde um grupo de possíveis objetivos está presente, e apenas um deles é verdadeiro. Assim, o algoritmo aplicado neste trabalho deve criar um caminho em que a chance deste modelo de identificação estar correto seja minimizada.

Este modelo trabalha com posições individuais no mapa, onde cada nodo vai ter uma distribuição de probabilidade de estar relacionado a qualquer um dos objetivos. Assim, pegando um agente em qualquer determinada posição do mapa, e considerando a sua posição inicial, é possível calcular a probabilidade dele estar se direcionando a cada um dos objetivos. Por exemplo na Figura 2.2, onde existem 2 possíveis objetivos representados por estrelas laranjas, e a posição inicial no diamante azul é conhecida. Logo, é possível assumir que a probabilidade de um agente na posição do X verde estar se direcionando ao objetivo da direita é mais alta do que o objetivo da esquerda.

Esta distribuição de probabilidade de cada objetivo para uma determinada posição é realizada através da aplicação de um cálculo de diferença de custo (Equação 2.3) em cada um dos objetivos sendo considerados, e comparando os resultados obtidos, onde quanto menor é o diferença de custo (*costdif(s,g,O)*), maior a probabilidade de um agente naquela posição estar

Figura 2.2 – Exemplo do modelo de reconhecimento



Fonte: Próprio autor

relacionada a aquele objetivo. Então, dada uma posição inicial no mapa e um grupo de objetivos, é possível calcular uma distribuição de probabilidade dos objetivos para cada posição do mapa. Este cálculo não é baseado nos movimentos anteriores do agente, já que eles são feitos apenas considerando a distância da posição atual até o objetivo, tal como descrito nas implementações apresentadas em (MASTERS; SEBASTIAN, 2017). A Equação 2.3 demonstra como o cálculo da diferença de custo é efetuado para um dos objetivos, onde:

- s representa a posição inicial do agente;
- g a posição do objetivo sendo considerado;
- O a posição atual do agente;
- *optc(s, O, g)* o custo ótimo da posição atual do agente até o objetivo, calculado com uma execução do algoritmo A*; e
- *optc(s, g)* representa o custo ótimo da posição inicial do agente até o objetivo, também calculado com a execução do algoritmo A*.

$$costdif(s,g,O) = optc(s,O,g) - optc(s,g) \tag{2.3}$$

Após aplicar este cálculo para todos os objetivos, é gerada a distribuição de probabilidade de uma determinada posição. Quanto menor a diferença de custo entre os caminhos que levam ao objetivo sendo considerado (*costdif(s, g, O)*), maior a probabilidade de aquela posição estar relacionada ao objetivo g. Com este modelo, é possível categorizar um passo da busca de caminhos (um nodo aberto pelo algoritmo de busca) como:

- **revelador**: se é mais provável o passo estar relacionado com o objetivo verdadeiro do que qualquer outro, ou seja, *costdif(s, g_r, O)* < *costdif(s, g, O)*, para todos $g \in G \backslash g_r$.
- **enganoso**: se o passo não for revelador, ou seja, a probabilidade da posição do mapa sendo analisada é maior para algum dos objetivos falsos do que para o objetivo verdadeiro.

Agora com um modelo em mente, é possível criar um caminho que subverte este, e é aqui que entra o trabalho de Masters e Sebastian (2017). Esse trabalho explora a aplicação prática destes conceitos na construção de caminhos enganosos, e é usado como base de implementação para este TCC. O trabalho propõem uma técnica para calcular caminhos enganosos em mapas *2D* com múltiplos objetivos onde a chance de um observador determinar o objetivo real de um agente a partir do seu caminho é minimizada. Nesta técnica, o algoritmo foca em enganar o observador entre 2 objetivos, o verdadeiro (g_r) e o objetivo falso mais próximo do verdadeiro (g'). Então, o primeiro passo deste processo é determinar qual objetivo falso é o g', comparando o custo de g_r até todos os objetivos e selecionando o menor. É importante ressaltar que um caminho jamais será enganoso em todos os seus passos pois eventualmente ele terá que convergir para o objetivo real. Assim, a função de um caminho enganoso é minimizar o número de passos que revelam o objetivo real.

É possível construir um caminho enganoso simplesmente indo até g' e depois se mover até g_r. Contudo, este caminho não vai ser natural para muitas aplicações e pode apresentar uma distância que é desnecessariamente alta. Partindo da ideia de que cada posição do mapa pode ser categorizada como enganosa ou reveladora, e que o agente deve eventualmente chegar ao seu objetivo real, o caminho necessariamente vai conter passos reveladores. Nessa mesma lógica, vai existir uma posição enganosa no mapa mais próxima do objetivo real do que todas outras, que é o último ponto enganoso possível (*Last Deceptive Point* - LDP)(MASTERS; SEBASTIAN, 2017). Por conseguinte, todos os passos do caminho a partir daquele ponto serão reveladores. Mais ainda, é possível categorizar um caminho como **fortemente enganoso** se todos os passos anteriores ao LDP são enganosos, e **fracamente enganoso** se o caminho apresenta passos reveladores antes do seu último passo enganoso (MASTERS; SEBASTIAN, 2017).

Como descrito por Masters e Sebastian (2017), para calcular o LDP é possível simplesmente fazer uma busca em profundidade até se encontrar uma posição enganosa. Porém, isso é computacionalmente excessivo, já que a verificação de engano não é computacionalmente negligível. Neste caso, é proposta uma alternativa baseada no raio de probabilidade máxima (Radius of Maximum Probability - RMP). Esse conceito determina um raio em volta de um objetivo em que se garante que não há nodos enganosos presentes. Se sabe que o LDP vai estar posicionado no meio do caminho ótimo entre g_r e g, e assim para encontrar a sua posição exata, vai se percorrer de passo a passo este caminho até se encontrar uma posição enganosa, começando na distância limite do RMP, onde se sabe que não têm posições enganosas.

Importante notar que este trabalho assume que o observador tem uma visão completa do mapa. Assim, o algoritmo age de forma enganosa em todos os passos do caminho sendo computado. Outro aspecto relevante a ser notado é que um caminho enganoso não é a mesma coisa que um caminho furtivo, pois o seu objetivo é enganar um observador de suas intenções, não despistá-lo. Neste caso, existe espaço para estudos que combinam os conceitos de caminhos enganosos e furtivos (Cai et al. (2020)), porém não é no escopo deste trabalho.

Tabela 2.1 – Resultados de testes de desempenho de algoritmos de busca.

	Custo Médio	Tempo Médio (s)
Algoritmo A*	215,9	0,208
Algoritmo enganoso, estratégia ds1	375,2	1,378
Algoritmo enganoso, estratégia ds2	245,2	1,997
Algoritmo enganoso, estratégia ds3	245,6	1,927
Algoritmo enganoso, estratégia ds4	248,7	1423,8

Fonte: Masters e Sebastian (2017)

Em Masters e Sebastian (2017), 4 diferentes estratégias para a criação de caminhos enganosos são apresentadas, cada uma resultando em caminhos diferentes e com tempos de processamento variados. As 4 estratégias estão descritas a seguir e exemplificadas na Figura 2.3.

- A primeira estratégia (denominada *ds1*) apenas combina um caminho entre o ponto inicial e o objetivo falso e um caminho entre o objetivo falso e o verdadeiro. Isso gera um caminho fortemente enganoso. Porém, é um caminho custoso e aparentemente artificial para um observador humano.

- A segunda estratégia (denominada *ds2*) combina um caminho entre o começo, o LDP calculado, e um caminho do LDP até o objetivo verdadeiro. Essa estratégia não garante um caminho fortemente enganoso.

- A terceira estratégia (denominada *ds3*) é similar com a segunda. Porém, o A* usado para ir até o LDP é alterado para priorizar passos mais enganosos. Isso é realizado alterando o custo dos nodos explorados para que valores de custo sejam multiplicados por 1.5 caso o custo heurístico (a distância Euclidiana da posição sendo avaliada até o objetivo verdadeiro) seja maior que a distância até o objetivo falso. Essa estratégia não garante um caminho fortemente enganoso.

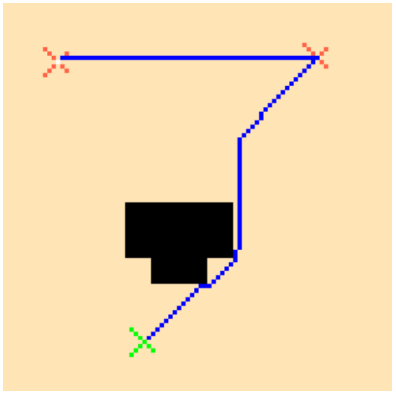
- Por último, a quarta estratégia (denominada *ds4*) utiliza o mesmo conceito da *ds2*. Porém, bloqueia qualquer posição reveladora do caminho até se chegar no LDP. Isso garante um caminho fortemente enganoso.

Nos testes de desempenho do trabalho original de Masters e Sebastian (2017), foram gerados 50 caminhos com cada um dos algoritmos atingindo os resultados apresentados na Tabela 2.1. O artigo não apresenta detalhes sobre o ambiente de teste. Então, o principal ponto de comparação para este trabalho é a proporcionalidade entre os diferentes algoritmos e valores de engano.

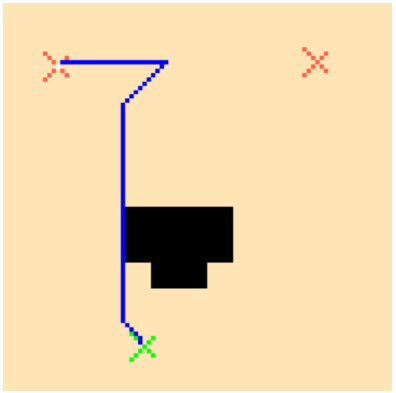
Como é possível observar, o algoritmo enganoso com a estratégia *ds4* apresenta um tempo de processamento muito elevado, visto que para cada passo do trajeto antes do LDP é necessário verificar se a posição é enganosa ou reveladora, e para isso é necessário calcular o

Figura 2.3 – Computações de caminhos enganosos de acordo com diferentes estratégias.

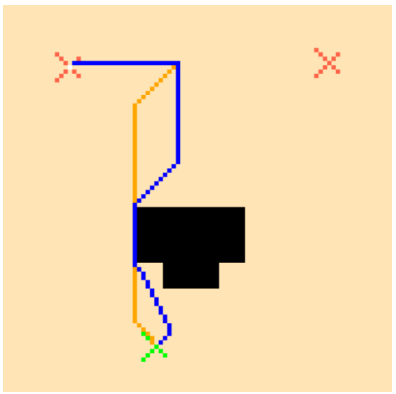
(a) Algoritmo enganoso, estratégia ds1



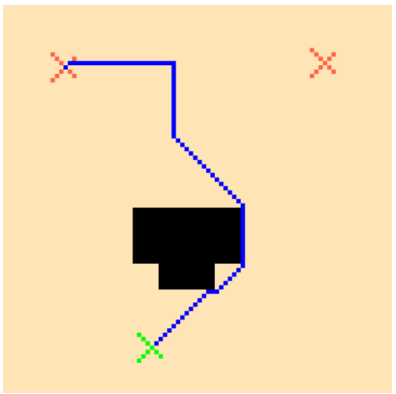
(b) Algoritmo enganoso, estratégia ds2



(c) Algoritmo enganoso, estratégia ds3



(d) Algoritmo enganoso, estratégia ds4



Fonte: Masters e Sebastian (2017)

custo ótimo de 2 caminhos (Equação 2.3). No trabalho original é teorizado que uma possível técnica para se combater o longo tempo de processamento desta estratégia é preparar previamente um *heatmap* de engano, já que dado uma posição inicial e um conjunto de objetivos, os valores de engano de cada posição do mapa são estáticos. Porém, o cálculo desse heatmap é muito lento (já que se calcula o engano em todas as posições do mapa), e se a posição inicial ou de um dos objetivos for alterada, o *heatmap* deve ser recalculado completamente.

2.4 TRABALHOS RELACIONADOS

Masters (2019) faz um estudo que serve como a base teórica para o trabalho de Masters e Sebastian (2017), demonstrando um modelo de engano que precisa apenas da posição inicial e posição atual do agente, sem necessitar informação prévias do caminho sendo calculado. O trabalho também aprofunda este modelo, como por exemplo o cálculo do RMP que é utilizado neste trabalho, ou a ideia de simulação (mostrar uma mentira) e dissimulação (esconder o real).

As ideias apresentadas no trabalho então são comparadas com o estado da arte de caminhos enganosos. Os resultados apresentados são positivos, necessitando menos informações e tempo de processamento que as alternativas comparadas.

Xu et al. (2020b) aprofunda o trabalho de Masters e Sebastian (2017), expandindo técnicas apresentadas e desenvolvendo novas técnicas para calcular caminhos enganosos. Em particular, o trabalho original considera apenas um dos objetivos falsos na hora de calcular o caminho enganoso. A versão proposta então considera múltiplos objetivos, e o método desenvolvido também apresenta um maior número de parâmetros, permitindo o usuário escolher um caminho com um maior nível de engano, resultando também em um caminho mais longo, aumentando a versatilidade do algoritmo. Partindo das propostas apresentadas em Xu et al. (2020b), Xu et al. (2020a) descreve uma otimização para as técnicas de busca de caminhos enganosos, com foco em mapas de grande escala para combater o crescimento exponencial do tempo de processamento. Isto é realizado a partir de uma solução híbrida entre escalonamento do mapa e abstração hierárquica. Isso resulta em um tempo de processamento reduzido com apenas uma pequena perda de engano no caminho resultante. Esses dois trabalhos demonstram algumas das formas que o trabalho de Masters e Sebastian (2017) já foi expandido, e também abre opções onde as técnicas desenvolvidas neste TCC podem ser aplicados.

A grande maioria dos trabalhos sobre caminhos enganosos lidam com cenários onde os elementos são estáticos. Porém, Cai et al. (2020) propõem um método de panejamento de caminho que considera informações dinâmicas no cenário, onde existe obstáculos não conhecidos no planejamento inicial do trajeto, e faz escolhas de como desviar destes enquanto se movimenta. Isto é realizado em duas etapas, primeiro é calculado um caminho enganoso que considera os elementos estáticos apenas, ou seja, a posição inicial do agente e os objetivos reais e falsos. Depois, a cada passo realizado o agente verifica se algum obstáculo é encontrado dentro do raio de detecção e faz escolhas de alterar o caminho caso julgar que vai ocorrer conflito entre o caminho e obstáculo. O principal resultado é um caminho com perfil enganoso que desvia de elementos dinâmicos considerados no deslocamento do agente pelo caminho.

Choi et al. (2012) apresenta um algoritmo de planejamento de caminhos global para mapas de relevo que considera 3 diferentes parâmetros: distância percorrida, tempo de viagem e gasto de energia. O método apresentado é baseado no algoritmo A* que é estendido para fazer os diferentes tipos de cálculos, alterando o custo de movimentação baseado no parâmetro sendo considerado. O resultado é um algoritmo que considera mais parâmetros físicos que estão presentes na vida real resultando em maior aplicabilidade, e é algo que pode ser explorado em caminhos enganosos e otimizados com redes neurais.

No contexto do projeto de pesquisa onde este TCC está inserido, Neisse et al. (2022) explora a aplicação de DNN como funções heurísticas para terrenos com elevação e inclinação, fazendo experimentos comparativos entre heurísticas aprendidas pelas DNN e heurísticas tradicionais. O treinamento da DNN foi feito com dados formados por caminhos calculados com o algoritmo A* usando a distância Euclidiana tridimensional como função heurística. Os

testes deste trabalho foram feitos em 3 diferentes mapas. Na execução do programa, esses mapas foram transformados em grafos direcionados. Os resultados obtidos apresentam uma redução significativa no tempo de cálculo dos caminhos, com apenas uma diferença de 0.3% nas distâncias/custos dos caminhos resultantes.

Similar ao trabalho do Neisse et al. (2022), Li et al. (2016) apresenta um algoritmo de busca de caminho heurística baseado no A* que utiliza uma DNN como um coeficiente multiplicador da função heurística. Porém, o fator multiplicador não considera os nodos da estrutura do mapa como é feito no trabalho do Neisse et al. (2022), além de não ter sido otimizado e testado para mapas com relevo e elevação.

Novamente no contexto do projeto de pesquisa onde este TCC está inserido, Chagas et al. (2022) propõem um algoritmo de busca de caminhos em terrenos com características topológicas, o *HPATheta**. Esse trabalho aborda problemas de suavidade em áreas com elevação e de desempenho em mapas de grande escala. O algoritmo é inspirado pelo algoritmo *Theta** (DANIEL et al., 2014) considerando a linha de visão do agente para a suavização do caminho e pelo algoritmo HPA* (BOTEA; MüLLER; SCHAEFFER, 2004) que utiliza compartimento hierárquico para reduzir o tempo de computação e o uso de memória. Visto que o trabalho de Xu et al. (2020a) demonstra a aplicação de técnicas de computação hierárquica para caminhos, o trabalho mostra como isso pode ser feito em terrenos com características topográficas.

Comparando os diferentes trabalhos apresentados nesta sessão, Xu et al. (2020b), Xu et al. (2020a) e Cai et al. (2020) trabalham com busca de caminhos enganosos de diversas formas, porém todos eles trabalham com terrenos planos, e apenas Xu et al. (2020a) da atenção para o desempenho das técnicas apresentadas. Já os trabalhos que lidam com a busca de caminho em terrenos com relevo, Choi et al. (2012) trabalha com parâmetros diferentes de apenas calcular o caminho de menor distância, Neisse et al. (2022) e Chagas et al. (2022) têm foco na otimização do processamento dos caminhos nestes terrenos. Já Li et al. (2016) e novamente Neisse et al. (2022) aplicam DNN para reduzir o tempo de processamento de caminhos.

De forma similar ao trabalho de Xu et al. (2020b), este TCC expande o trabalho de Masters e Sebastian (2017). Porém, ao invés de adicionar novas funcionalidades ao trabalho, as técnicas apresentadas são aplicadas e analisadas em terrenos com relevo, tópico ainda não explorado na computação de caminhos enganosos. Como em Xu et al. (2020a), este trabalho também busca otimizar o processamento de algoritmos enganosos. Porém, este problema é tratado com a aplicação de aprendizado de máquina, algo previamente não explorado na literatura de caminhos enganosos.

3 BUSCA DE CAMINHOS ENGANOSOS EM TERRENOS COM TOPOGRAFIA E OTIMIZADOS COM DNNS

Esta sessão descreve como a pesquisa e implementação deste trabalho foram realizadas. Primeiro, diferentes mapas com relevo que foram utilizados nos testes dos algoritmos (Sessão 3.1). Em seguida, conceitos de caminhos enganos foram aplicados nos mapas com características topográficas. Além disso, o comportamento resultante destes testes são apresentados (Sessão 3.2). Então, o trabalho detalhada como foram aplicadas as DNN para otimizar o processamento destes caminhos e como estas redes foram treinadas (Sessão 3.3). O próximo capítulo (Capítulo 4) apresenta e discute os resultados dos testes realizados nas implementações descritas nesta sessão.

As implementações desenvolvidas neste trabalho foram realizadas na linguagem *Python* [1], usando como base o *software p4* (Sebastian Sardina, Peta Masters, 2021). Esse software serve como ferramenta de visualização dos algoritmos de busca de caminhos implementados. Ele também contém uma implementação do algoritmo de busca de caminhos enganosos do trabalho de (MASTERS; SEBASTIAN, 2017), que foi ajustado para os objetivos deste trabalho.

3.1 MAPAS UTILIZADOS

Para a realização dos testes deste TCC, foram utilizados três mapas obtidos a partir de cortes (segmentos) de uma imagem de satélite fornecida pela Nasa. Essa imagem segue o Modelo Digital de Elevação (MDE), que é uma representação das altitudes da superfície topográfica agregada aos elementos geográficos existentes. Este modelo utiliza o formato de imagem *Tag Image File Format* (TIF), onde a altura do terreno é representada pela cor dos pixeis.

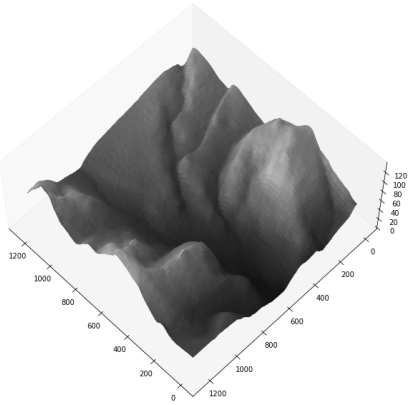
Os três mapas utilizados neste trabalho têm uma resolução de 100x100 *pixels*, onde cada pixel representa uma distância de 12,5 metros. Os mapas originalmente foram selecionados de forma a conter diferentes densidade de variação de altura, o que é relevante para este TCC para expor o impacto que terrenos mais montanhosos terão nas métricas de execução. Quando esses mapas são carregados em memória, essas alturas são convertidas para uma matriz do mesmo tamanho do mapa contendo a informação de elevação em cada ponto.

As Figuras 3.1, 3.2 e 3.3 apresentam uma visualização 3D e histograma da distribuição da elevação dos mapas com relevo selecionados para o desenvolvimento deste trabalho.

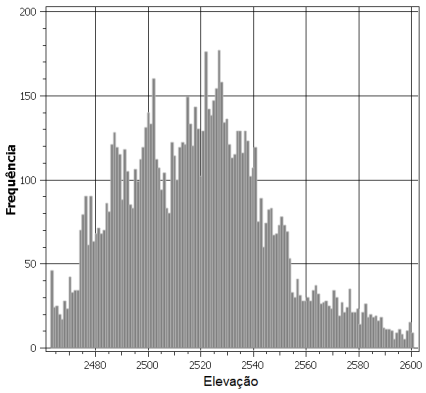
[1] https://www.python.org/

Figura 3.1 – Terreno e distribuição da elevação do mapa 1.

(a) Terreno



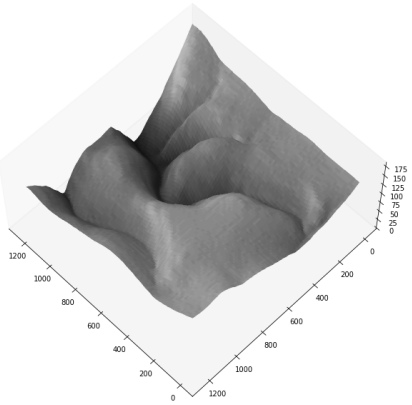
(b) Distribuição da elevação



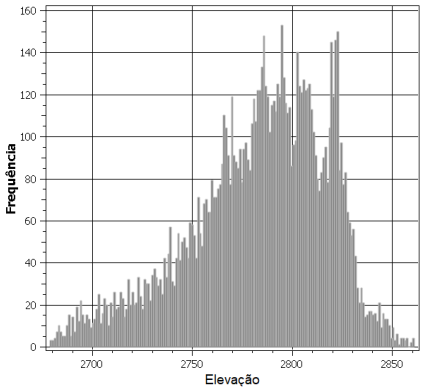
Fonte: Próprio autor

Figura 3.2 – Terreno e distribuição da elevação do mapa 2.

(a) Terreno



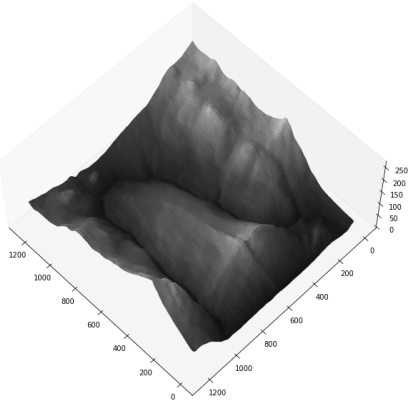
(b) Distribuição da elevação



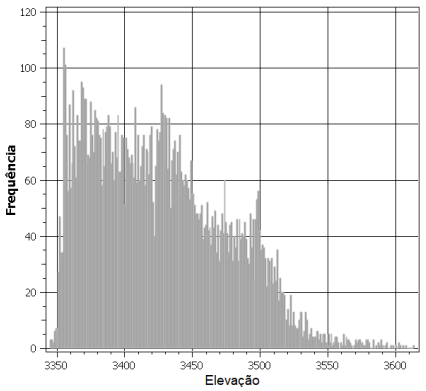
Fonte: Próprio autor

Figura 3.3 – Terreno e distribuição da elevação do mapa 3.

(a) Terreno



(b) Distribuição da elevação



Fonte: Próprio autor

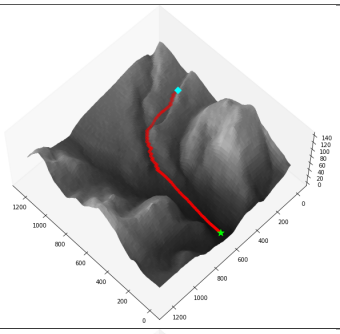
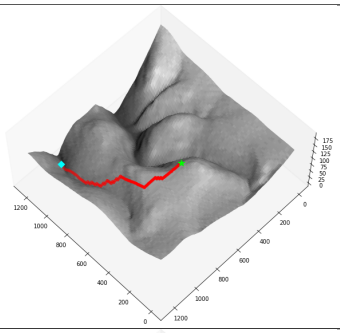
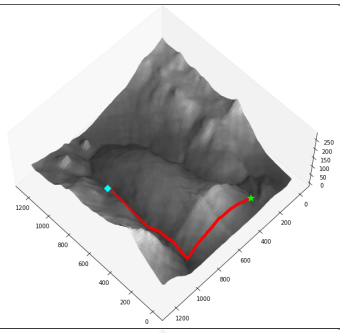
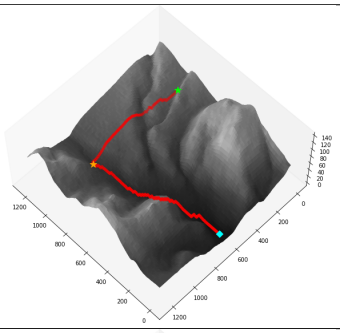
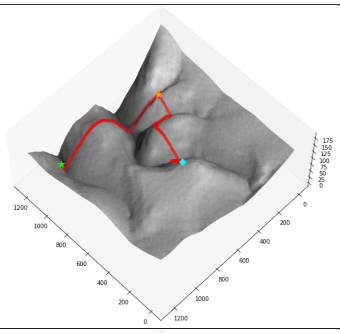
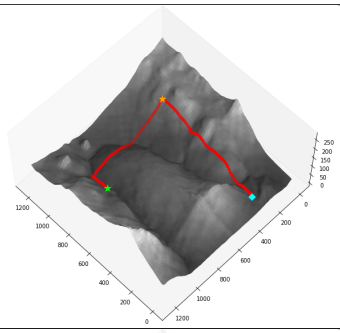
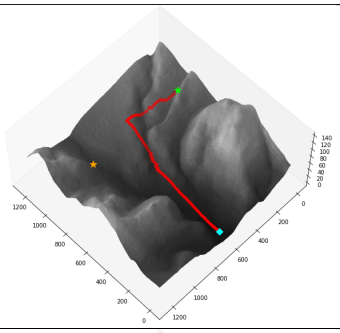
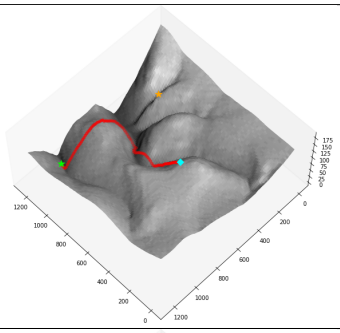
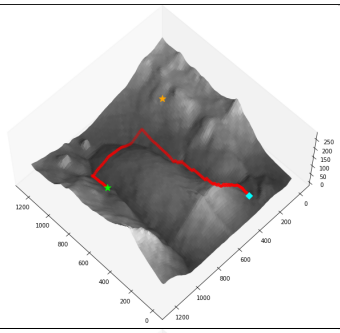
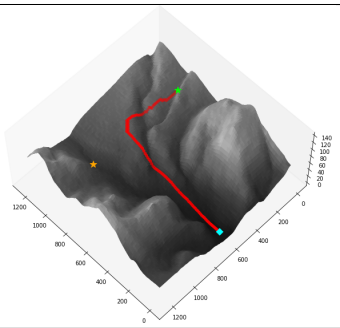
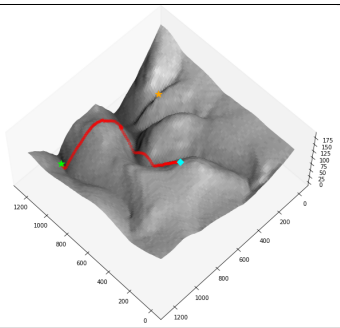
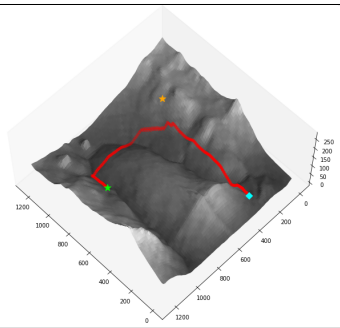
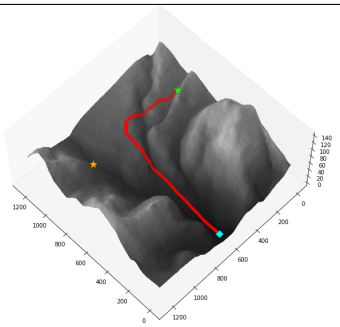
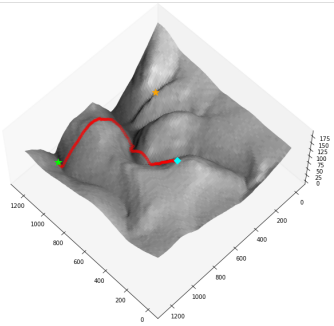
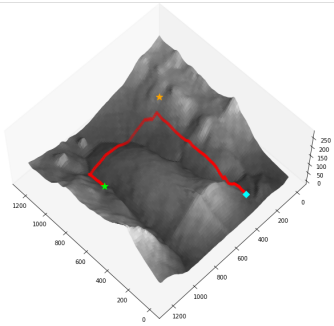
3.2 CAMINHO ENGANOSO EM MAPAS COM RELEVO

Para a realização de testes, as quatro versões do algoritmo de busca de caminhos enganosos apresentadas na Sessão 2.3 (MASTERS; SEBASTIAN, 2017) foram adaptadas para considerar as características topológicas do terreno. Nos algoritmos originais, os custos usados para determinar se uma posição é reveladora ou enganosa são calculados com o algoritmo A* (Algoritmo 1). Para adaptar o conceito de passo enganoso para um terreno com elevação, o algoritmo A* foi modificado para utilizar a distância Euclidiana tridimensional (que considera diferença de altura entre nodos consecutivos). Essa função de distância então computa o custo do passo local de busca *g(n)* bem como computa a função heurística *h(n)* do algoritmo de busca.

A Tabela 3.1 demonstra exemplos nos três diferentes mapas usados de caminhos calculados pelas quatro técnicas enganosas ajustadas neste trabalho, incluindo também o caminho calculado pelo algoritmo A* padrão para fins de comparação. Nas Figuras, o diamante azul representa a posição inicial do agente, a estrela laranja representa o objetivo falso, e a estrela verde o objetivo verdadeiro. Assim como implementado neste trabalho, a lógica do algoritmo de busca enganoso permite a presença de mais de um objetivo falso. Porém, para fins da computação deste caminho, apenas um destes objetivos é considerado, onde apenas um objetivo falso foi incluído nas imagens exemplificadas e testes realizados neste trabalho.

Como pode ser observado nas imagens da Tabela 3.1, um caminho enganoso pode ser alterado significativamente quando o relevo do terreno é considerado no processo de busca. Isso pode ser notado quando esse caminho é comparado com caminho ótimo calculado pelo algoritmo A*. Análises mais detalhadas sobre isso são apresentadas na Sessão 4.1.

Tabela 3.1 – Caminhos gerados pelas diferentes estratégias de busca de caminhos testadas neste trabalho.

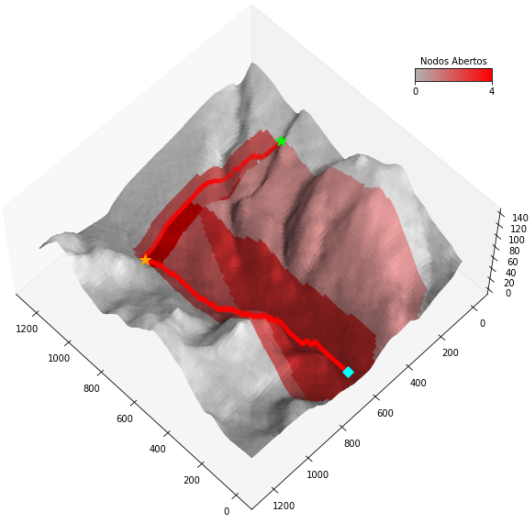
	Mapa 1	Mapa 2	Mapa 3
A*			
ds1			
ds2			
ds3			
ds4			

Fonte: Próprio autor

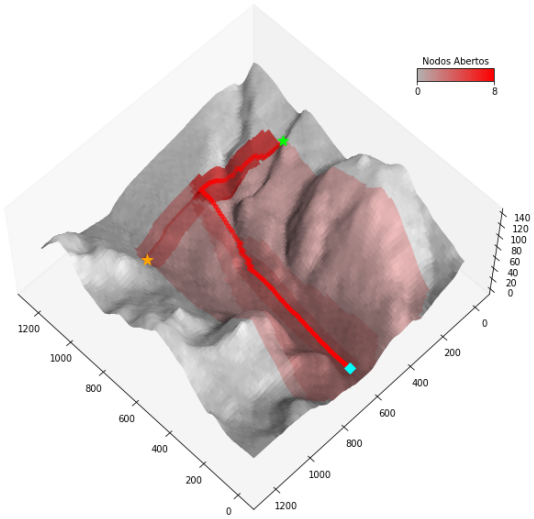
Figura 3.4 apresenta imagens com *heatmaps* de nodos expandidos de cada uma das versões dos algoritmos implementados. Quanto mais vermelho o ponto no mapa, mais vezes aquele nodo foi aberto durante o cálculo do caminho. Importante, cada uma das imagens apresenta uma escala própria de cores (com nuvens de cores representando valores em diferentes escalas), onde o valor mais alto de qualquer um dos nodos abertos no processo de busca está descrito na legenda da imagem. Como pode ser observado na Figura 3.4, todas as versões dos algoritmos de busca abrem uma quantia significativa de nodos, principalmente o algoritmo enganoso com estratégia *ds4*. Isso vai servir como um ponto inicial para a comparação das estratégias implementadas, permitindo visualizar o efeito da rede neural na escolha de nodos realizadas por esses algoritmos de busca.

Figura 3.4 – Nodos expandidos durante a execução de diferentes estratégias enganosas realizadas pelo algoritmo de busca implementado.

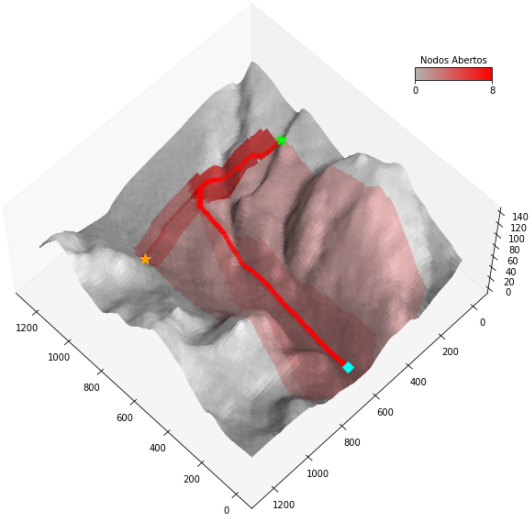
(a) Algoritmo enganoso, estratégia ds1



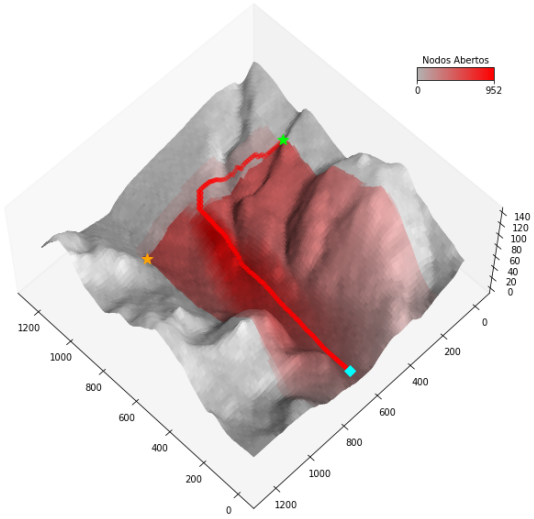
(b) Algoritmo enganoso, estratégia ds2



(c) Algoritmo enganoso, estratégia ds3



(d) Algoritmo enganoso, estratégia ds4



Fonte: Próprio autor

3.3 DNN PARA PREVISÃO DE CUSTO DE CAMINHOS

A partir de testes preliminares realizados neste trabalho, foi possível analisar que o tempo de processamento destes caminhos, especialmente no que diz respeito ao algoritmo enganoso com estratégia ds4, é relativamente alto. Esse tempo elevado de processamento é um impeditivo para o emprego destes algoritmos enganosos em aplicações que exigem respostas em tempo real. A maioria deste tempo de processamento está relacionada com os cálculos de caminhos ótimos utilizando o algoritmo A*. Esses caminhos são aplicados tanto na criação do trajeto final entre os pontos considerados quanto na determinação do engano de uma posição no mapa como demonstrado na Equação 2.3. Logo, um dos focos de pesquisa deste trabalho é otimizar essas computações de caminhos.

Como forma de otimização, este trabalho analisa a efetividade da aplicação de uma DNN para realizar predições de custo de caminhos entre 2 pontos em um mapa com relevo. A DNN é modelada com base no trabalho de Neisse et al. (2022) e Neisse (2021), onde foi utilizada para calcular a função heurística do algoritmo A* adaptado para terrenos com características topográficas. Mais especificamente, é utilizada a versão denominada h_{subAll} citada naquele trabalho, que treina e aplica uma mesma DNN para 3 diferentes mapas, visto que essa abordagem apresentou os melhores resultados e versatilidade.

Neste trabalho, duas possíveis aplicações para o emprego de DNN são exploradas:

1. Técnica onde a função heurística do A* é substituída pela DNN: A técnica apresentada no trabalho de Neisse et al. (2022), onde a DNN é utilizada no local da função heurística do algoritmo A* para otimizar as escolhas de nodos a serem explorados por este algoritmo. Ou seja, $h(n) = dnn(n)$;

2. Técnica onde a DNN é usada na estimativa do engano das posições no mapa, realizada durante a execução das diferentes estratégias enganosas testadas neste trabalho: essa técnica aplica a DNN para prever o custo do caminho ótimo entre dois pontos do mapa. Essa abordagem é computada conforme a Equação 2.3, a qual é usada para determinar o engano em uma posição no mapa, substituindo a função *optc* por um retorno da DNN. Esta aplicação apresenta um potencial de ganho de velocidade de processamento para o algoritmo de busca enganoso, visto que substituiu um cálculo de um caminho inteiro do algoritmo A* por um acesso a DNN treinada.

Assim como descrito na Sessão 4.2, portanto, se testou e analisou o uso da DNN neste trabalho com as seguintes abordagens:

- Abordagem de emprego de DNN denominada **aplicação completa**. Essa abordagem explora a aplicação simultânea da técnica 1 e a técnica 2 citadas anteriormente, com o intuito de minimizar o tempo de execução do algoritmo enganoso;

- Abordagem de emprego de DNN denominada **aplicação parcial**. Assim como descrito

na Sessão 4.3, apenas a técnica 1 citada acima é explorada, com o objetivo de obter um caminho mais próximo daquele calculado sem o uso da DNN.

Implementação e treinamento das DNN foram realizados utilizando as bibliotecas TensorFlow (ABADI et al., 2015) e Keras (CHOLLET et al., 2015). O modelo apresenta 7 neurônios na camada de entrada, um neurônio na camada de saída que utiliza uma função linear como função de ativação, e onze camadas ocultas que utilizam a função de ativação ReLU, visto que os valores nos conjuntos de dados não foram normalizados. A quantidade de neurônios em cada camada apresenta a seguinte configuração: [7,500,100,200,300,400,500,400,300,200,100,20,1].

Em cada entrada no dataset de treinamento, existem 3 entradas para a posição inicial do caminho, considerando a coordenada x, a coordenada y, e a altura deste ponto, mais 3 entradas para a posição final do caminho. A sétima entrada indica o identificador do mapa sendo utilizado, representado por um número inteiro de 1 a 3 (visto que o trabalho foi desenvolvido e testado em três diferentes mapas com topografia). O atributo de saída retorna o custo estimado do caminho entre esses pontos (custo ótimo do caminho considerando o relevo do mapa pré-computado pela execução do algoritmo de busca).

O conjunto de dados utilizado para o treinamento da DNN contém caminhos entre possíveis combinações de posições iniciais e finais para os 3 mapas onde a DNN é aplicada, onde cada um destes caminhos são acompanhados com o caminho ótimo. Esses dados foram gerado utilizando um programa que faz uma busca de profundidade completa para cada posição do mapa, o qual foi acelerado utilizando unidades de processamento gráfico. Como critério de parada do treinamento, foi determinado 2000 épocas, e se atingiu um MAPE de 0,3143, o que significa uma precisão de 96,85% na DNN treinada.

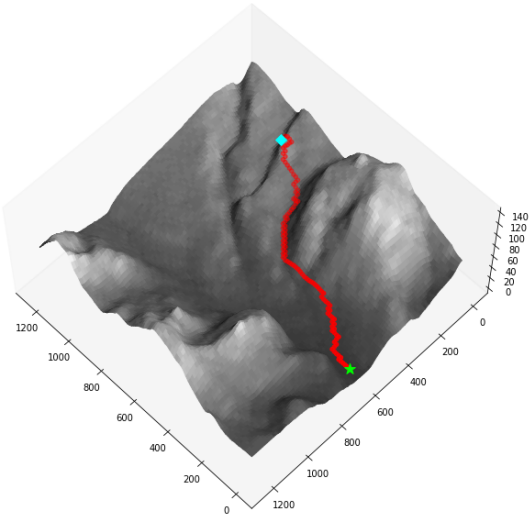
Com a DNN treinada, ela foi aplicada no programa desenvolvido neste trabalho. Quando um caminho é calculado, a DNN é alimentada com um conjunto de entradas de todas as posições do mapa até os dois objetivos sendo considerados e também até o LDP. Então a DNN realiza as predições para todas as entradas e os resultados são armazenados em um dicionário que pode ser acessado a partir de valores de posição inicial e final. Assim, todo o tempo de processamento relacionado a DNN se origina da predição, e qualquer consulta depois do dicionário ter sido gerado apresenta um custo desprezível. Diferente da técnica de otimização do algoritmo de busca enganoso com estratégia *ds4* sugerida no trabalho de Masters e Sebastian (2017), onde é necessário calcular previamente um *heatmap* de engano para cada conjunto de posição inicial e final no mapa, a DNN depois de treinada é aplicada em qualquer caminho naquele mapa, precisando apenas realizar a predição da DNN para cada caminho.

Na Figura 3.5 está demonstrado exemplos de execuções no mapa 1 de cada uma das versões dos algoritmos implementados com a aplicação completa da DNN. Comparando esses caminhos com as figuras demonstradas na Tabela 3.1, é possível perceber que existe uma pequena diferença entre os caminhos gerados com e sem DNN, porém, o formato geral do caminho é mantido, com exceção do algoritmo A* que tomou um caminho diferente mas com um custo similar. Esses resultados são explorados em maior detalhe no Capítulo 4 deste trabalho.

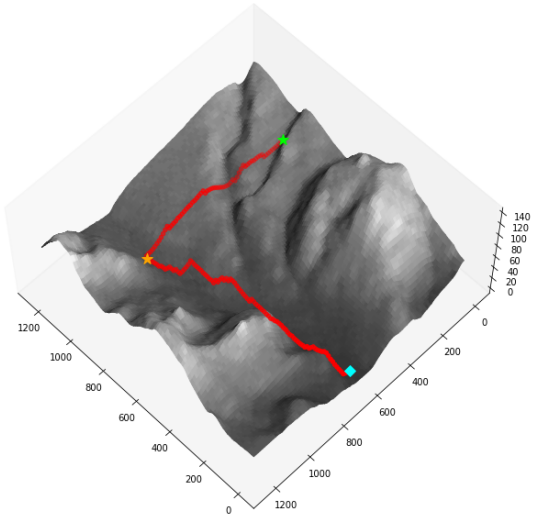
Figura 3.6 demonstrada os nodos expandidos pelo algoritmo durante o processamento do caminho com a aplicação completa da DNN.

Figura 3.5 – Execução dos algoritmos de busca considerando características topográficas e aplicação completa da DNN.

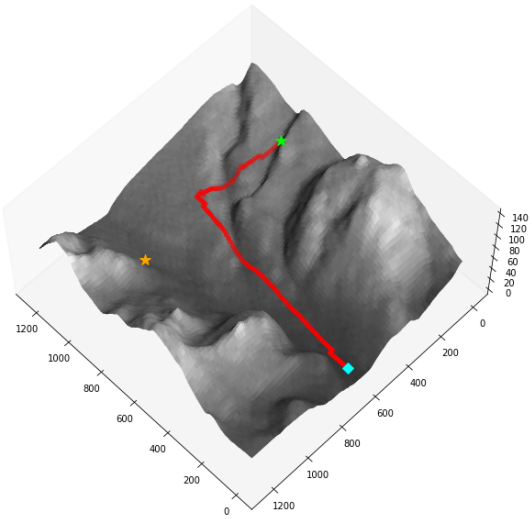
(a) Algoritmo A*



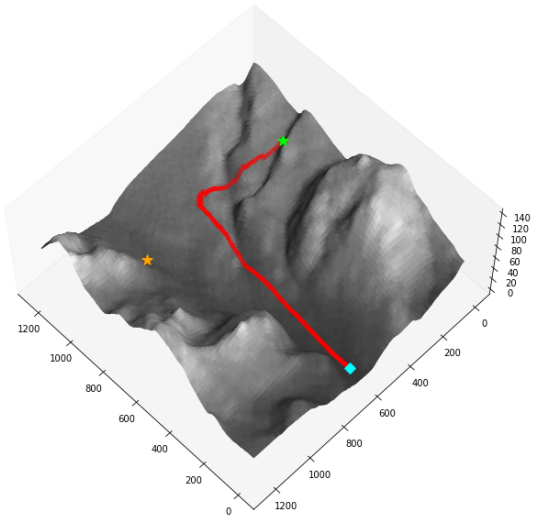
(b) Algoritmo enganoso, estratégia ds1



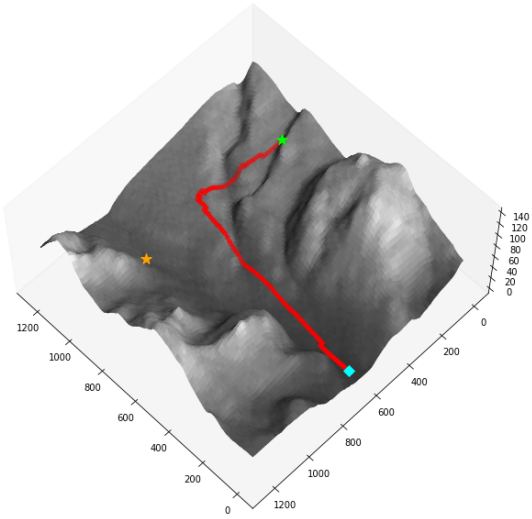
(c) Algoritmo enganoso, estratégia ds2



(d) Algoritmo enganoso, estratégia ds3



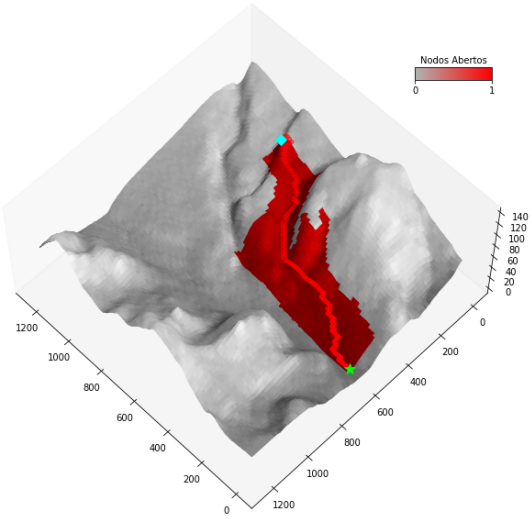
(e) Algoritmo enganoso, estratégia ds4



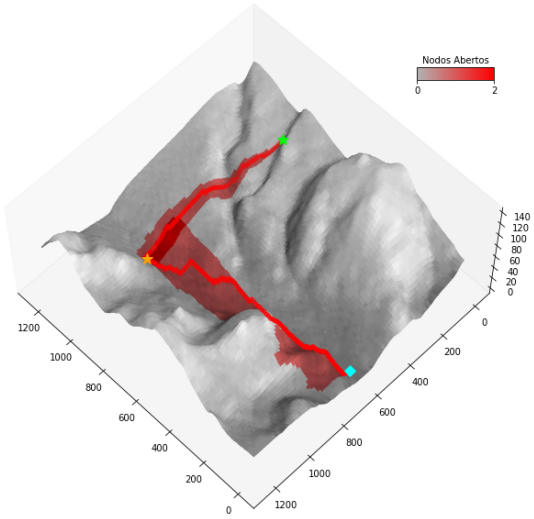
Fonte: Próprio autor

Figura 3.6 – Nodos expandidos durante a execução do algoritmo de busca com aplicação completa da DNN.

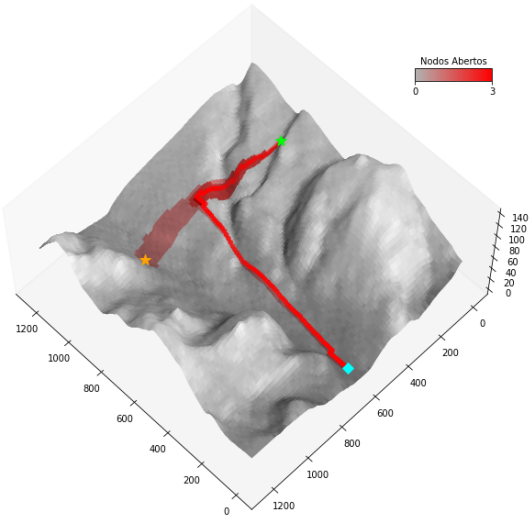
(a) Algoritmo A*



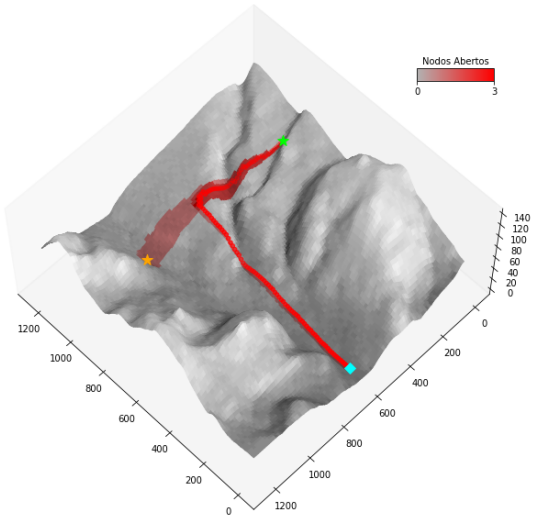
(b) Algoritmo enganoso, estratégia ds1



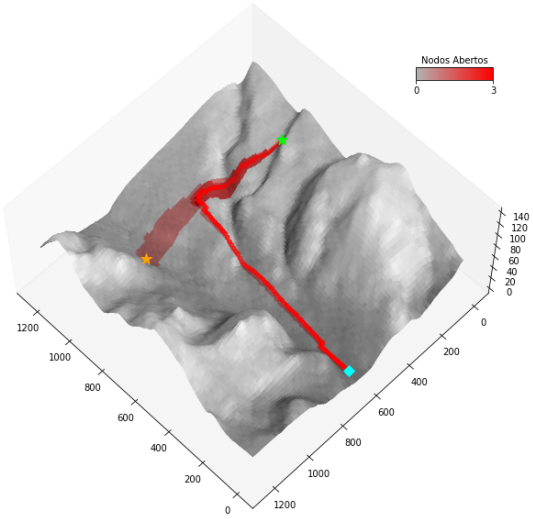
(c) Algoritmo enganoso, estratégia ds2



(d) Algoritmo enganoso, estratégia ds3



(e) Algoritmo enganoso, estratégia ds4



Fonte: Próprio autor

4 EXPERIMENTOS E RESULTADOS

Os experimentos realizados neste trabalho tem o objetivo de analisar a diferença entre as técnicas de busca selecionadas quando aplicadas em terrenos com relevo, e também analisar o desempenho dos algoritmos com e sem o uso de DNN, e o impacto que isso tem nos caminhos computados. As diferenças dos resultados em 3 diferentes mapas (sessão 3.2) também são avaliadas para determinar o impacto das diferentes topografias dos terrenos sobre os resultados. Os algoritmos analisados são as 4 versões de caminhos enganosos apresentadas no Capítulo 3, além do algoritmo A*, que é usado de base de comparação. Esses algoritmos foram todos analisados em 3 diferentes contextos de emprego. Primeiro, sem o uso de DNN (Sessão 4.1). Segundo, com a aplicação completa da DNN (Sessão 4.1). Terceiro, com a aplicação parcial da DNN (Sessão 4.3). Em particular, as aplicações de DNN são detalhadas na Sessão 3.3.

Com o intuito de realizar uma análise das técnicas em questão, as seguintes métricas foram usadas:

- Tempo de execução do algoritmo de busca;
- Número de nodos expandidos na execução do algoritmo de busca;
- Custo/Distância do caminho retornado pelo algoritmo de busca;
- Densidade de passos enganosos no caminho retornado pelo algoritmo de busca.

Para efetuar análises comparativas de forma estatística, modelos de Regressão Linear Generalizadas (GLM) (MCCULLAGH; NELDER, 1983) são usados. Esses modelos são uma generalização dos modelos de regressão linear que permitem variáveis com distribuições diferentes da normal. O modelo de regressão linear generalizada utilizado neste trabalho está apresentado na Equação 4.1.

$$log(\mu) = \beta_0 + \beta_1 X + \beta_2 D_1 + \beta_3 D_2 + \beta_4 D_3 + \beta_5 D_4 \tag{4.1}$$

Onde μ é a média da distribuição da variável resposta, X é a variável explicativa e D_1, D_2, D_3, D_4 são as variáveis *dummy*, cada uma representando uma das estratégias que serão comparadas com o método base (o algoritmo A*). Um nível de significância de $\alpha = 0,1\%$ foi definido, e as hipóteses nulas $\mathbb{H}_0$ e alternativa $\mathbb{H}_1$ foram definidas como $\beta_i = 0$ ou $\beta_i \neq 0$, para i = 0, 1, 2 e 3. Se $\alpha > valor - p$, então $\mathbb{H}_0$ não é rejeitada. $\beta_1 > 0$ significa que a variável resposta é diretamente proporcional a variável explicativa. $\beta_{i+1} > 0$, $\beta_{i+1} = 0$ e $\beta_{i+1} < 0$ para $i = 1,2,3,4$, significam, respectivamente, que a função heurística associada a D_i é mais eficiente, equivalente, e menos eficiente que o algoritmo A* que é usado como base de comparação.

Para que estes algoritmos possam ser analisado estatisticamente utilizando a técnica apresentada acima, é necessário que se tenha um *dataset* de execuções com os resultados das

métricas sendo avaliadas. Para este fim, é necessário um conjunto de cenários de aplicação (posição inicial e conjunto de objetivos) que serão executados para que esse *dataset* possa ser formado, e assim foram gerados 1000 cenários distintos para cada um dos mapas, cada um com uma posição inicial, um objetivo verdadeiro, e um objetivo falso. A posição destes parâmetros foi escolhida de forma aleatória, seguindo regras de distanciamento mínimo e máximo entre as posições, com uma alta variação de alcance entre os pontos considerados para simular diversos contextos de aplicação. Após gerados, todos os cenário de cada mapa foram executados por cada um dos algoritmos testados neste trabalho, uma vez sem usar DNN, outra vez com a aplicação completa da DNN, e também com a aplicação parcial da DNN. A quantia de cenários foi escolhida de acordo com o tempo de execução do algoritmo mais lento entre eles, o algoritmo enganoso com estratégia *ds4*, que sem o uso de DNN tem um tempo de processamento médio de 66 segundos por caminho. O ambiente computacional usado na execução dos testes é apresentado na Tabela 4.1.

Tabela 4.1 – Ambiente de execução dos testes.

CPU	Intel(R) Core(TM) i5-4670K
CPU Núcleos	4
CPU Frequência	3.40GHz
GPU	NVIDIA GeForce GTX 970
GPU Memória	4GB
Memória RAM	16GB
Sistema Operacional	Windows 10

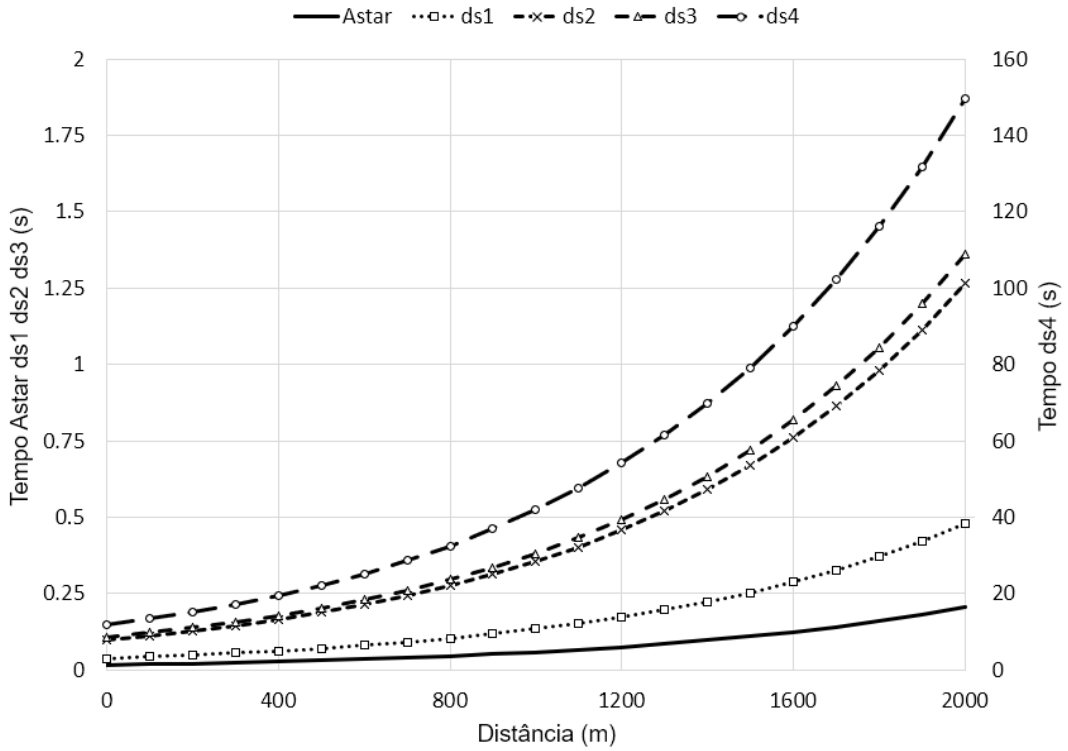
Fonte: Próprio autor

4.1 CAMINHOS ENGANOSOS EM TERRENOS COM CARACTERÍSTICAS TOPOGRÁFICAS

Os resultados dos testes executados sem o uso de DNN e as análises correspondentes são apresentados a seguir. Primeiro, as métricas de execução no Mapa 1 são apresentadas e analisadas, com um foco na diferença entre as técnicas enganosas testadas. Depois, os resultados nos Mapas 2 e 3 são apresentados, onde é analisada a variação dos resultados entre os 3 mapas. O objetivo é analisar como as técnicas enganosas interagem com diferentes terrenos e o impacto que isso tem nas métricas de execução testadas.

Nos gráficos de tempo por distância (Figuras 4.1, 4.3, 4.5) e nodos expandidos por distância (Figuras 4.2, 4.4, 4.6), um eixo vertical adicional na direita do gráfico é utilizado para o algoritmo enganoso com estratégia *ds4*. Isso permite que os resultados de todos os algoritmos testados possam ser melhor representados em uma escala comparável e visível.

Figura 4.1 – Mapa 1 (Sem DNN): Tempo X Distância.



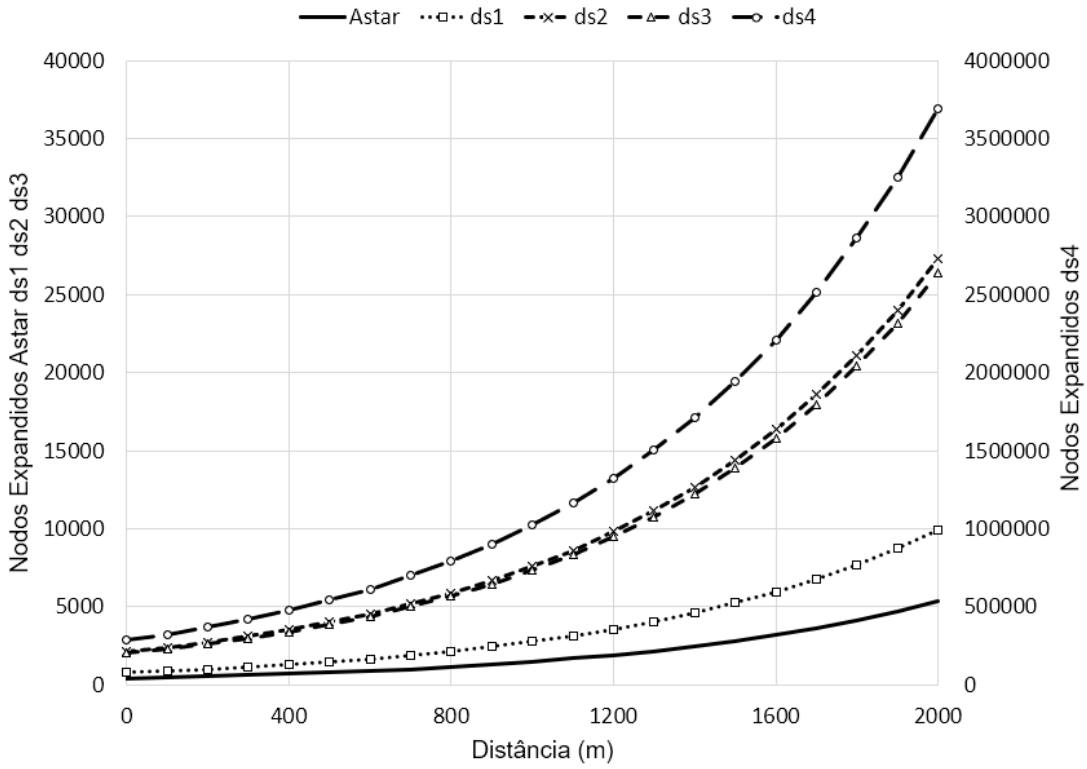
Fonte: Próprio autor

Tabela 4.2 – Resultados estatísticos, Mapa 1 (Sem DNN): Tempo X Distância.

	Estimativa	Desvio Padrão	Valor t	$P(>\|t\|)$
Intercepto	-4,118e+00	3,127e-02	-131,67	<2e-16
Algoritmo A*	1,272e-03	2,323e-05	54,76	<2e-16
Algoritmo enganoso, estratégia ds1	8,344e-01	3.151e-02	26,48	<2e-16
Algoritmo enganoso, estratégia ds2	1,808e+00	2,785e-02	64,92	<2e-16
Algoritmo enganoso, estratégia ds3	1,882e+00	2,785e-02	67,58	<2e-16
Algoritmo enganoso, estratégia ds4	6,582e+00	2,822e-02	233,25	<2e-16

Fonte: Próprio autor

Figura 4.2 – Mapa 1 (Sem DNN): Nodos Expandidos X Distância.



Fonte: Próprio autor

Tabela 4.3 – Resultados estatísticos, Mapa 1 (Sem DNN): Nodos Expandidos X Distância.

	Estimativa	Desvio Padrão	Valor t	$P(>\|t\|)$
Intercepto	6.018e+00	3.110e-02	193.47	<2e-16
Algoritmo A*	1.281e-03	2.307e-05	55.53	<2e-16
Algoritmo enganoso, estratégia ds1	6.250e-01	3,123e-02	20,01	<2e-16
Algoritmo enganoso, estratégia ds2	1,634e+00	2,764e-02	59,11	<2e-16
Algoritmo enganoso, estratégia ds3	1,600e+00	2,764e-02	57,88	<2e-16
Algoritmo enganoso, estratégia ds4	6,542e+00	2,799e-02	233,74	<2e-16

Fonte: Próprio autor

Tabela 4.4 – Porcentagem de diferença de tempo de execução e nodos expandidos em relação ao algoritmo A* (Sem DNN).

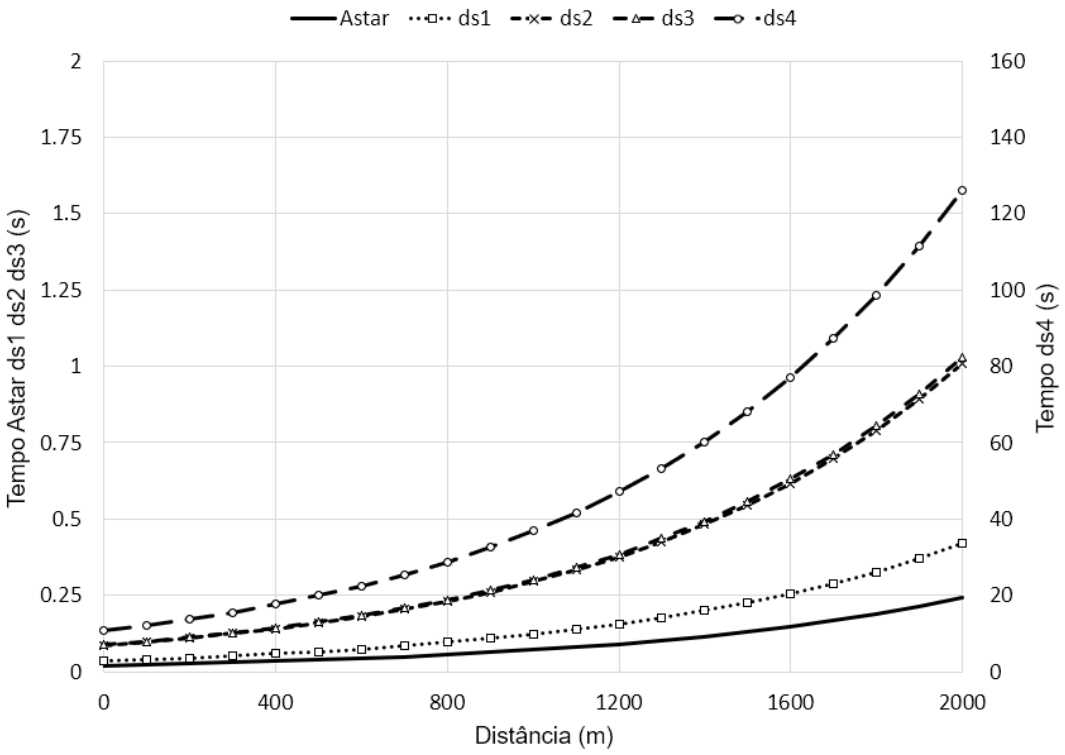
	Tempo de execução	Nodos Expandidos
Algoritmo enganoso, estratégia ds1	+130.34%	+86.82%
Algoritmo enganoso, estratégia ds2	+509.82%	+412.43%
Algoritmo enganoso, estratégia ds3	+556.66%	+395.30%
Algoritmo enganoso, estratégia ds4	+72098.18%	+69267.25%

Fonte: Próprio autor

Tabela 4.4 apresenta as diferenças de cada estratégia em relação ao cálculo do caminho

ótimo realizado pelo A*. Em geral, a maioria do tempo de processamento dos algoritmos testados está relacionado com a exploração de nodos do mapa; logo, não é uma surpresa que existe uma proporcionalidade entre as métricas analisadas. A diferença de tempo de processamento entre os algoritmos difere da apresentada em Masters e Sebastian (2017) (Tabela 2.1). Porém, a ordem de performance das estratégias é similar, onde o algoritmo A* é o mais rápido. Acima disso está o algoritmo enganoso com estratégia *ds1*, com valores aproximadamente dobrando aqueles apresentados pelo algoritmo A*. O algoritmo enganoso com as estratégias *ds2* e *ds3* tem um tempo de processamento similar para essas estratégias, e o algoritmo enganoso com estratégia *ds4* apresenta um tempo de processamento extremamente elevado comparado com as demais estratégias enganosas.

Figura 4.3 – Mapa 2 (Sem DNN): Tempo X Distância.



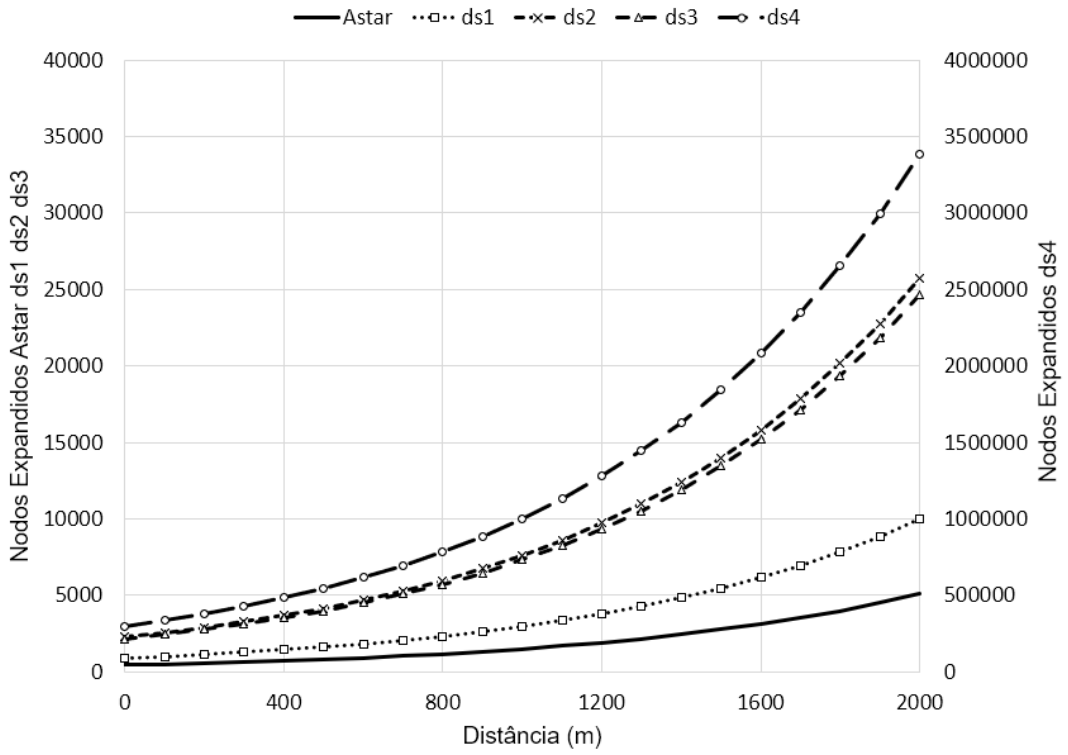
Fonte: Próprio autor

Tabela 4.5 – Resultados estatísticos, Mapa 2 (Sem DNN): Tempo X Distância.

	Estimativa	Desvio Padrão	Valor t	$P(>\|t\|)$
Intercepto	-3.880e+00	3.084e-02	-125.83	<2e-16
Algoritmo A*	1.231e-03	2.216e-05	55.57	<2e-16
Algoritmo enganoso, estratégia ds1	5,474e-01	3,085e-02	17,74	<2e-16
Algoritmo enganoso, estratégia ds2	1,427e+00	2,787e-02	51,20	<2e-16
Algoritmo enganoso, estratégia ds3	1,448e+00	2,788e-02	51,95	<2e-16
Algoritmo enganoso, estratégia ds4	6,256e+00	2,822e-02	221,71	<2e-16

Fonte: Próprio autor

Figura 4.4 – Mapa 2 (Sem DNN): Nodos Expandidos X Distância.



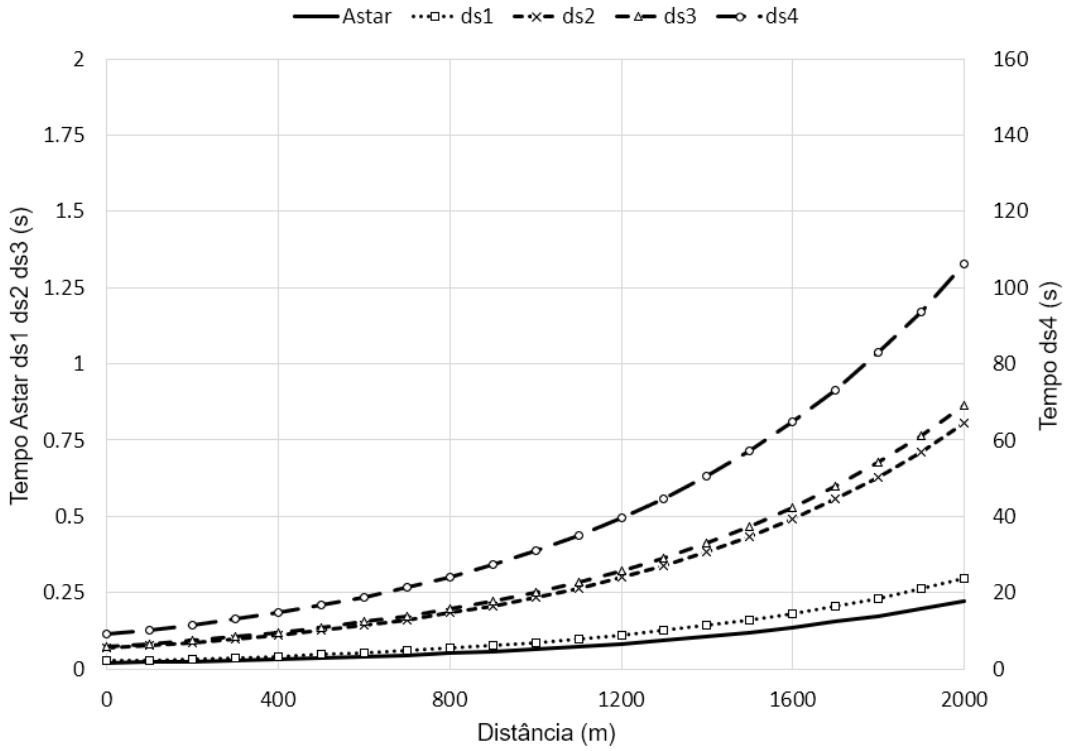
Fonte: Próprio autor

Tabela 4.6 – Resultados estatísticos, Mapa 2 (Sem DNN): Nodos Expandidos X Distância.

	Estimativa	Desvio Padrão	Valor t	$P(>\|t\|)$
Intercepto	6.098e+00	3.024e-02	201.63	<2e-16
Algoritmo A*	1.217e-03	2.176e-05	55.92	<2e-16
Algoritmo enganoso, estratégia ds1	6,792e-01	3,038e-02	22,36	<2e-16
Algoritmo enganoso, estratégia ds2	1,623e+00	2,742e-02	59,19	<2e-16
Algoritmo enganoso, estratégia ds3	1,582e+00	2,742e-02	57,70	<2e-16
Algoritmo enganoso, estratégia ds4	6,503e+00	2.777e-02	234,17	<2e-16

Fonte: Próprio autor

Figura 4.5 – Mapa 3 (Sem DNN): Tempo X Distância.



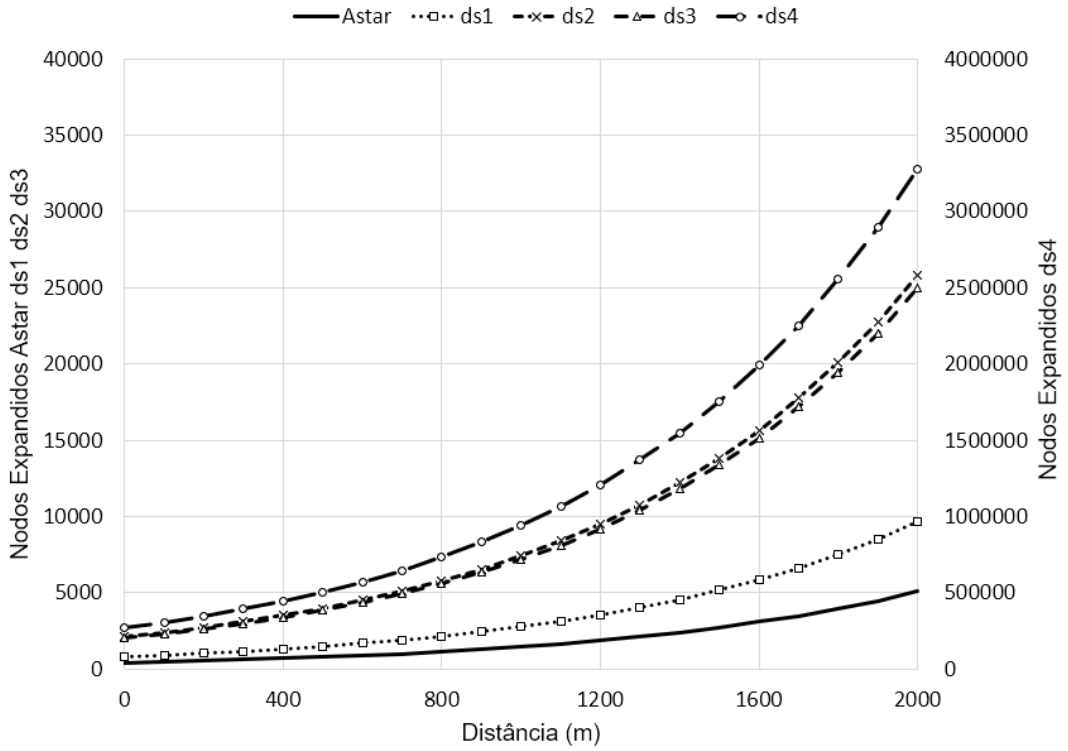
Fonte: Próprio autor

Tabela 4.7 – Resultados estatísticos, Mapa 3 (Sem DNN): Tempo X Distância.

	Estimativa	Desvio Padrão	Valor t	$P(>\|t\|)$
Intercepto	-3.971e+00	3.026e-02	-131.255	<2e-16
Algoritmo A*	1.235e-03	2.174e-05	56.812	<2e-16
Algoritmo enganoso, estratégia ds1	2,851e-01	3,056e-02	9,328	<2e-16
Algoritmo enganoso, estratégia ds2	1,284e+00	2,740e-02	46,851	<2e-16
Algoritmo enganoso, estratégia ds3	1,356e+00	2,740e-02	49,498	<2e-16
Algoritmo enganoso, estratégia ds4	6,165e+00	2,781e-02	221,671	<2e-16

Fonte: Próprio autor

Figura 4.6 – Mapa 3 (Sem DNN): Nodos Expandidos X Distância.



Fonte: Próprio autor

Tabela 4.8 – Resultados estatísticos, Mapa 3 (Sem DNN): Nodos Expandidos X Distância.

	Estimativa	Desvio Padrão	Valor t	$P(>\|t\|)$
Intercepto	6,039e+00	2.936e-02	205,66	<2e-16
Algoritmo A*	1,247e-03	2,111e-05	59,09	<2e-16
Algoritmo enganoso, estratégia ds1	6,402e-01	2,967e-02	21,58	<2e-16
Algoritmo enganoso, estratégia ds2	1,625e+00	2,658e-02	61,15	<2e-16
Algoritmo enganoso, estratégia ds3	1,592e+00	2,658e-02	59,88	<2e-16
Algoritmo enganoso, estratégia ds4	6,469e+00	2,697e-02	239,89	<2e-16

Fonte: Próprio autor

Como apresentado na Sessão 3.1, os Mapas 2 e 3 têm menos variação de relevo do que o Mapa 1, e os testes de execução realizados sobre estes mapas utilizam o mesmo conjunto de entradas e parâmetros que aqueles realizados no Mapa 1. A partir dos resultados estatísticos apresentados nas Tabelas 4.5 e 4.7, se observou uma redução no tempo de processamento e de nodos expandidos nos algoritmos enganosos. Já que a única diferença entre as execuções nos diferentes mapas é o seu terreno, ou seja, a variação de altura entre os nodos do mapa, se observa uma relação entra a quantia de variações de relevo e os suas métricas de execução. Essa relação é lógica já que terrenos mais planos vão resultar em caminhos mais retos com menos desvios nos nodos explorados. Isso pode ser principalmente observado no algoritmo enganoso com estratégia *ds4*.

Para analisar a efetividade (o engano) das diferentes estratégias enganosas nos terrenos com relevo testados, foi registrada a quantia de passos reveladores de todos os caminhos computados. Lembrando, o objetivo destes algoritmos é minimizar a quantia de passos reveladores. As estratégias enganosas são comparadas em relação ao número de passos do caminho ótimo do LDP até o objetivo real, que representa o menor número de passos reveladores possível de um caminho computado. A média da diferença percentual nos 3 mapas testados é apresentada na Tabela 4.9.

Tabela 4.9 – Média da diferença do número de passos reveladores de cada estratégia de busca em relação ao valor ótimo (Sem DNN).

	Mapa 1	Mapa 2	Mapa 3
Algoritmo A*	+76,06%	+82,25%	+84,89%
Algoritmo enganoso, estratégia ds1	+0,00%	+0,00%	+0,00%
Algoritmo enganoso, estratégia ds2	+29,60%	+33,88%	+36,48%
Algoritmo enganoso, estratégia ds3	+33,00%	+37,02%	+38,51%
Algoritmo enganoso, estratégia ds4	+0,00%	+0,00%	+0,00%

Fonte: Próprio autor

O algoritmo enganoso com estratégias *ds1* e *ds4* são fortemente enganosos mesmo em terrenos com relevo, ou seja, apresentam o número mínimo de passos reveladores. A estratégia enganosa *ds2* apresentou resultados de engano superiores a estratégia *ds3*, porém os caminhos resultantes da estratégia *ds3* se aproximam mais daqueles calculados pela estratégia *ds4*.

Outra métrica importante a ser considerada é o custo (distância) dos caminhos computados. Para isso, foi calculada a diferença percentual de custo entre os algoritmos enganosos e o custo ótimo calculado pelo algoritmo A*. A média dos resultados de custo obtidos para cada um dos algoritmos em cada um dos mapas testados é apresentada na Tabela 4.10.

Tabela 4.10 – Média da diferença de custo de cada estratégia enganosa comparada com o algoritmo A* (Sem DNN).

	Mapa 1	Mapa 2	Mapa 3
Algoritmo enganoso, estratégia ds1	+112,42%	+114,49%	+112,19%
Algoritmo enganoso, estratégia ds2	+22,12%	+24,36%	+23,94%
Algoritmo enganoso, estratégia ds3	+22,52%	+25,17%	+24,69%
Algoritmo enganoso, estratégia ds4	+22,93%	+25,76%	+25,59%

Fonte: Próprio autor

Mesmo que o algoritmo enganoso com estratégia *ds1* apresente ótimos resultados de tempo de execução e engano, o seu custo elevado limita a aplicação real. Além disso, pode ficar claro para um observador lógico que após o agente chegar no objetivo falso e continuar o movimento até outro objetivo, um caminho enganoso está sendo tomado. Os algoritmos enganosos com estratégias *ds2*, *ds3* e *ds4* apresentam valores relativamente similares entre si. Isso coloca em evidência que para muitas aplicações, a diferença do valor de engano entre

as estratégias calculadas na Tabela 4.9 nem sempre significa que os caminhos resultantes vão ser muito diferentes, e que para um observador humano essas diferenças podem não ser tão perceptíveis. O algoritmo enganoso com estratégia *ds3* se aproxima do algoritmo enganoso com *ds4* em termos de trajeto computado, com um tempo de processamento muito similar ao algoritmo enganoso com estratégia *ds2*.

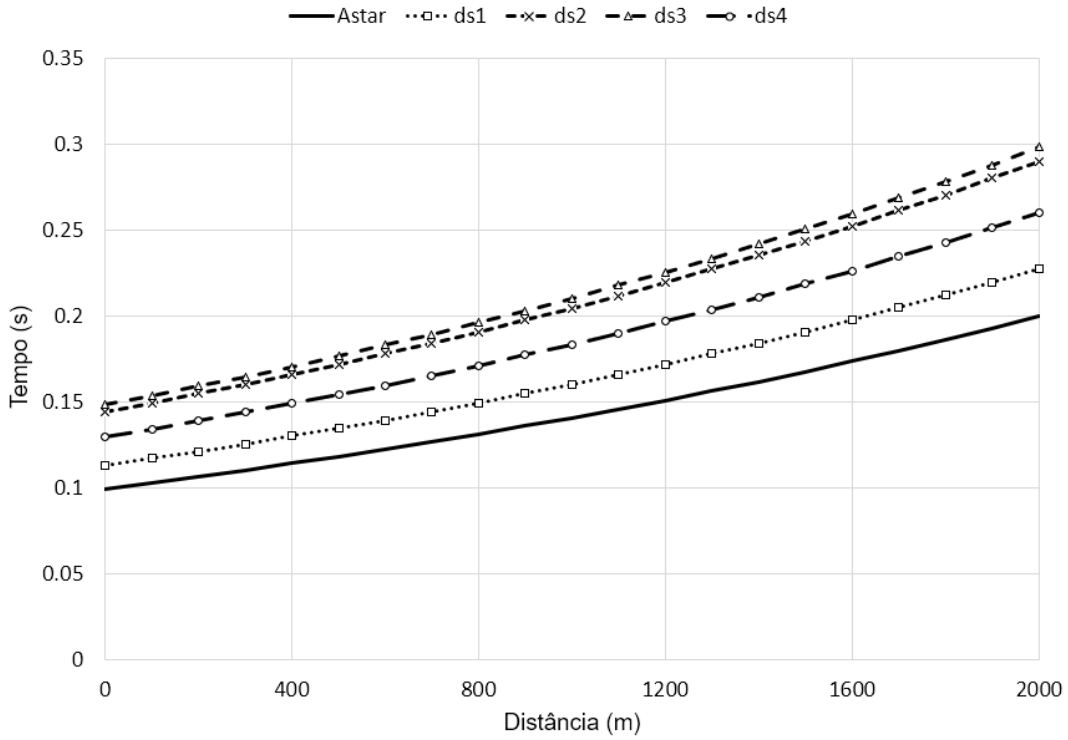
Considerando os resultados apresentados nesta sessão e os exemplos demonstrados na Tabela 3.1, é possível perceber que as estratégias enganosas podem ser aplicadas com bons resultados em terrenos com relevo, apresentando métricas de execução em linha com aquelas encontradas no trabalho de Masters e Sebastian (2017), além de caminhos lógicos que consideram a variação do relevo nos trajetos calculados.

4.2 APLICAÇÃO COMPLETA DA DNN EM CAMINHOS ENGANOSOS

Os resultados apresentados nesta sessão dizem respeito à aplicação completa da DNN, ou seja, a DNN é usada como heurística do algoritmo A* para o cálculo de caminhos, e essa mesma DNN também é usada para estimar o custo do caminho entre 2 posições do mapa (com o objetivo de determinar o quão enganosa é uma posição sendo avaliada) - ver definição de aplicação completa na Sessão 3.3. A ideia principal é analisar o uso de DNN na otimização do algoritmo A*, que por sua vez é usado na computação do algoritmo enganoso nas suas diferentes estratégias testadas. Inicialmente, é apresentado o impacto que a aplicação completa da DNN teve na métricas de tempo de processamento (Figura 4.11) e nodos abertos 4.12).

Para os testes de tempo de execução dos algoritmos de busca usando a DNN, está sendo considerada o tempo de predição do modelo da DNN, que é executado para cada caminho calculado. Além disso, está sendo desconsiderado o tempo de carregamento do arquivo do modelo de DNN (aproximadamente 1 segundo), já que esse modelo só é carregado uma única vez no início do programa para todos os testes.

Figura 4.7 – Mapa 1 (DNN Completa): Tempo X Distância.



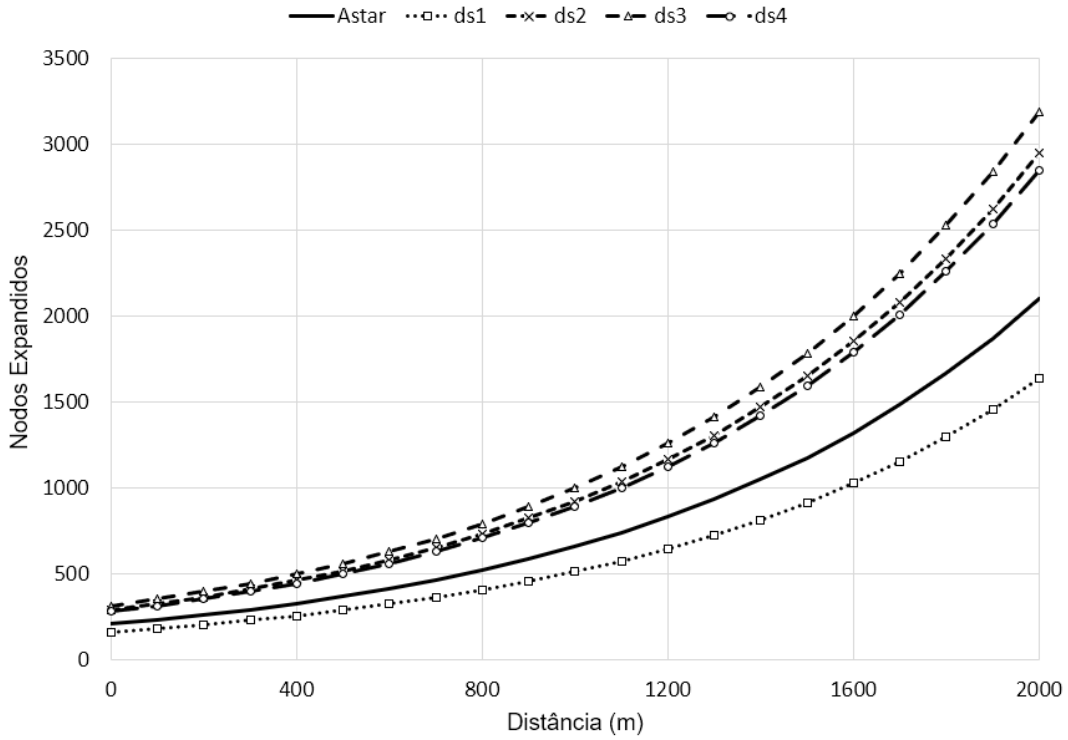
Fonte: Próprio autor

Tabela 4.11 – Resultados estatísticos, Mapa 1 (DNN Completa): Tempo X Distância.

	Estimativa	Desvio Padrão	Valor t	$P(>\|t\|)$
Intercepto	-2.311e+00	1.320e-02	-175.012	<2e-16
Algoritmo A* (Heurística DNN)	3.493e-04	9.844e-06	35.487	<2e-16
Algoritmo enganoso, estratégia ds1	1.32e-01	1.509e-02	-9.577	<2e-16
Algoritmo enganoso, estratégia ds2	3.740e-01	1.319e-02	28.349	<2e-16
Algoritmo enganoso, estratégia ds3	4.017e-01	1.317e-02	30.492	<2e-16
Algoritmo enganoso, estratégia ds4	2.656e-01	1.327e-02	20.012	<2e-16

Fonte: Próprio autor

Figura 4.8 – Mapa 1 (DNN Completa): Nodos Expandidos X Distância.



Fonte: Próprio autor

Tabela 4.12 – Resultados estatísticos, Mapa 1 (DNN Completa): Nodos Expandidos X Distância.

	Estimativa	Desvio Padrão	Valor t	$P(>\|t\|)$
Intercepto	5,326e+00	2,897e-02	183,87	<2e-16
Algoritmo A* (Heurística DNN)	1,162e-03	2,022e-05	57,46	<2e-16
Algoritmo enganoso, estratégia ds1	8.53e-02	2,928e-02	-8,62	<2e-16
Algoritmo enganoso, estratégia ds2	3,379e-01	2,647e-02	12,77	<2e-16
Algoritmo enganoso, estratégia ds3	4,160e-01	2,642e-02	15,74	<2e-16
Algoritmo enganoso, estratégia ds4	3,035e-01	2,652e-02	11,44	<2e-16

Fonte: Próprio autor

Tabela 4.13 – Porcentagem de diferença de tempo de execução e nodos expandidos em relação ao algoritmo A* (DNN Completa).

	Tempo de execução	Nodos Expandidos
Algoritmo enganoso, estratégia ds1	+14.11%	+8.90%
Algoritmo enganoso, estratégia ds2	+45.35%	+40.20%
Algoritmo enganoso, estratégia ds3	+49.55%	+51.59%
Algoritmo enganoso, estratégia ds4	+30.42%	+35.46%

Fonte: Próprio autor

Comparando os tempos de execução com e sem o uso de DNN, uma grande redução de

tempo de processamento é observada. Observando os valores da estimativa do algoritmo A* na Tabela 4.2 (Sem DNN), que determina o crescimento do tempo de execução em relação a distância, os algoritmos apresentam um crescimento de tempo de 1273.05% entre a distância de 0 e 2000. Na Tabela 4.11 (DNN Completa), analisando essa mesma métrica, o crescimento é de apenas 201.09% para a mesma faixa de distância. Este crescimento mais linear é desejável para aplicações em mapas de grande escala.

O número de nodos expandidos também sofreu uma grande redução, com valores aproximadamente 9 vezes mais baixos do que a versão dos algoritmos sem o uso de DNN. Porém, o crescimento do número de nodos expandidos em relação a distância percorrida continua similar, expondo outro benefício do uso da DNN. A abertura de nodos tem um impacto menor no tempo de processamento do que na execução do algoritmo de busca sem o uso de DNN.

Como observado na Tabela 4.13, a aplicação completa da DNN aproxima o tempo de execução das diferentes estratégias enganosas. O algoritmo enganoso com estratégia *ds4*, em específico, foi o mais afetado pela aplicação completa da DNN, com um tempo de execução médio mais de 300 vezes menor, e uma quantia de nodos expandidos 1.200 vezes menor do que a original. Devido a natureza das DNN, esse ganho de desempenho acabar causando desvios no trajeto quando esse caminho retornado é comparado a versão do mesmo algoritmo sem o uso de DNN. Para determinar o nível desta perda de otimização no custo do caminho calculado, análises do engano e do custo destes caminhos foram realizadas.

Para determinar o desvio do trajeto dos caminhos causado pela aplicação da DNN na computações heurísticas desenvolvidas pelo algoritmo A*, calculou-se a diferença percentual do custo dos caminhos. Isso permitiu comparar os resultados das versões dos algoritmos enganosos com e sem DNN. Equação 4.2 é usada para calcular o percentual de diferença de custo de cada caminho (C_{dif}), onde C_{dnn} é o custo do caminho calculado usando a DNN, C é o custo do caminho calculado pelo algoritmo sem o uso de DNN (o valor ideal), e a função *Abs* retorna o valor absoluto (positivo), para que diferenças de custo superiores e inferiores sejam consideradas igualmente. Então, é feita a média aritmética entre os resultados de cada algoritmo para cada mapa testado. Esses resultados são apresentados na Tabela 4.14.

$$C_{dif} = Abs(C_{dnn} - C)/C \tag{4.2}$$

Apenas comparar os custos dos caminhos não providencia uma análise da diferença dos trajetos resultantes, visto que dois caminhos com trajetos diferentes podem apresentar o mesmo custo. Porém, providencia uma forte sugestão da diferença entre estas estratégias de busca testadas.

O algoritmo $A*$ e o algoritmo enganoso com estratégia *ds1* tiveram diferenças de custo relativamente pequenas. Porém, é possível observar uma grande discrepância de custo nos algoritmos enganosos com estratégias *ds2*, *ds3* e *ds4*. Observando caminhos individuais, foi possível perceber que muitos dos caminhos calculados por essas estratégias apresentaram uma diferença abaixo de 1%. Porém, outros caminhos tiveram uma diferença maior que 150%. Isto é

Tabela 4.14 – Média da diferença de custo entre os caminhos retornados pelos algoritmos sem DNN e os algoritmos com aplicação completa da DNN.

	Mapa 1	Mapa 2	Mapa 3
Algoritmo A* (Heurística DNN)	0,76%	0,69%	1,46%
Algoritmo enganoso, estratégia ds1	1,04%	1,30%	1,75%
Algoritmo enganoso, estratégia ds2	24,64%	25,64%	28,96%
Algoritmo enganoso, estratégia ds3	24,32%	25,95%	28,74%
Algoritmo enganoso, estratégia ds4	27,91%	28,56%	32,77%

Fonte: Próprio autor

devido ao cálculo da posição do LDP, onde pequenas diferenças de predição de custo causadas pelo uso da DNN podem resultar em posições do LDP bem diferentes. Já que as estratégias enganosas se baseiam em se locomover até o LDP, o custo dos seus caminhos são diretamente afetados por estes desvios.

Para calcular o engano, utilizando o mesmo método apresentado na sessão anterior, o número de passos reveladores dos caminhos calculados são comparados com a quantia mínima possível de passos enganosos (o número de passos do LDP calculado sem o uso de DNN até o objetivo real). A média desses resultados é apresentada na Tabela 4.20.

Tabela 4.15 – Média da diferença do número de passos reveladores de cada estratégia de busca em relação ao valor mínimo possível (DNN Completa).

	Mapa 1	Mapa 2	Mapa 3
Algoritmo A* (Heurística DNN)	+68,19%	+81,27%	+75,05%
Algoritmo enganoso, estratégia ds1	+18,99%	+19,57%	+15,77%
Algoritmo enganoso, estratégia ds2	+46,05%	+45,22%	+39,69%
Algoritmo enganoso, estratégia ds3	+36,52%	+41,14%	+38,00%
Algoritmo enganoso, estratégia ds4	+33,79%	+33,17%	+22,95%

Fonte: Próprio autor

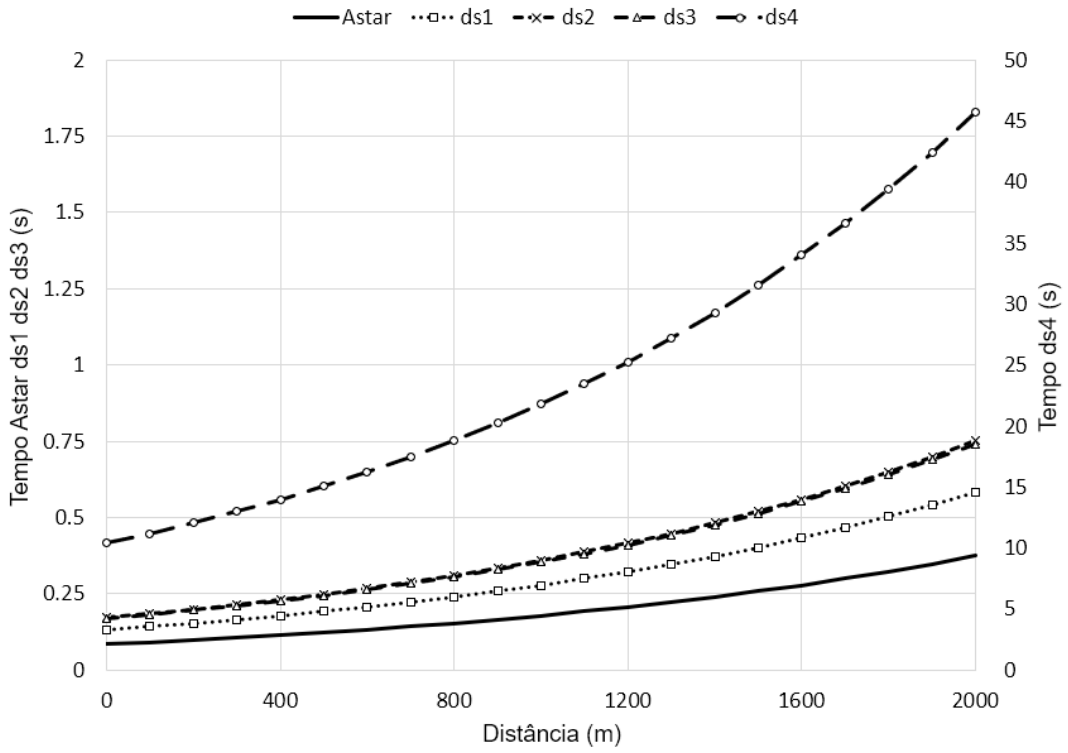
Comparando as Tabelas 4.9 e 4.15, é possível perceber um aumento de passos reveladores, ou seja, uma perda de engano, em todas as estratégias enganosas quando a aplicação completa da DNN é usada. O algoritmo enganoso com estratégia *ds3* obteve as menores perdas de engano em relação a versão deste algoritmo sem DNN. Porém, o algoritmo enganoso com a estratégia *ds4* ainda apresenta resultados superiores, e como visto na Tabela 4.13, com um tempo de execução similar as demais estratégias testadas.

A grande variação de custo encontrada na aplicação completa da DNN pode limitar a sua aplicabilidade em situações onde se deseja encontrar caminhos enganosos com baixos custos. Desta forma, é proposto na Sessão 4.3 formas alternativas de aplicar esta mesma DNN, onde não se atinge a mesma redução de tempo de processamento observada na aplicação completa da DNN, porém é possível alcançar um nível maior de precisão de custo.

4.3 APLICAÇÃO PARCIAL DA DNN

Esta sessão analisa as métricas de teste para a aplicação parcial da DNN nos algoritmos de busca. Isso significa que a DNN é apenas usada como heurística do algoritmo A*; porém, diferente da aplicação completa apresentada na sessão anterior deste trabalho, os cálculos de custos ótimos (*optc*) são realizados a partir do algoritmo A* (auxiliados pela DNN) ao invés de serem retornados diretamente pela DNN. Uma predição de custo ótimo imprecisa exerce um impacto muito maior no trajeto final do que um valor heurístico impreciso durante a execução do algoritmo A*. O intuito desta aplicação é apenas utilizar a DNN em locais onde a sua influencia é limitada, resultando em cálculos de custos ótimos com menores desvios dos valores reais do que aqueles estimados com a aplicação completa da DNN, e assim calculando posições de LDP mais próximas daquelas encontradas pelos algoritmos sem o uso DNN. Isso permite aliviar as deficiências encontradas na aplicação completa da DNN nos algoritmos de busca.

Figura 4.9 – Mapa 1 (DNN Parcial): Tempo X Distância.



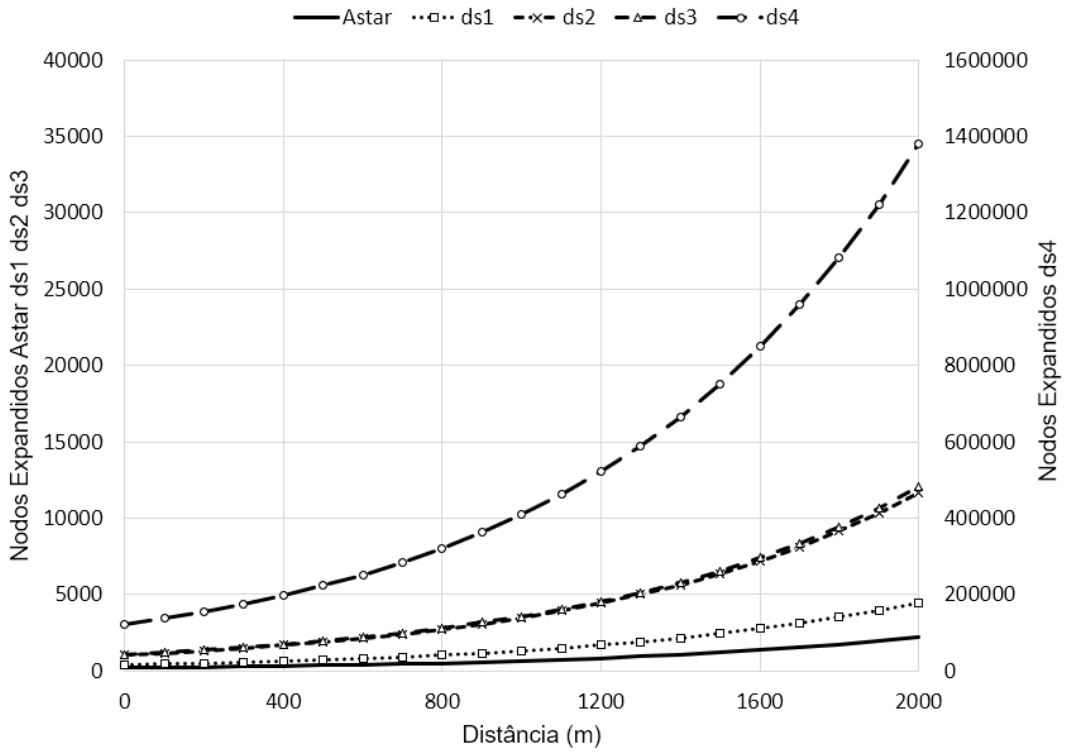
Fonte: Próprio autor

Tabela 4.16 – Resultados estatísticos, Mapa 1 (DNN Parcial): Tempo X Distância.

	Estimativa	Desvio Padrão	Valor t	$P(>\|t\|)$
Intercepto	-2,464e+00	3,088e-02	-79,803	<2e-16
Algoritmo A* (Heurística DNN)	7,397e-04	2,381e-05	31,068	<2e-16
Algoritmo enganoso, estratégia ds1	4,34e-01	1,509e-02	31,858	<2e-16
Algoritmo enganoso, estratégia ds2	6,995e-01	3,078e-02	22,729	<2e-16
Algoritmo enganoso, estratégia ds3	6,858e-01	3,076e-02	22,295	<2e-16
Algoritmo enganoso, estratégia ds4	4,806e+00	3,196e-02	150,387	<2e-16

Fonte: Próprio autor

Figura 4.10 – Mapa 1 (DNN Parcial): Nodos Expandidos X Distância.



Fonte: Próprio autor

Tabela 4.17 – Resultados estatísticos, Mapa 1 (DNN Parcial): Nodos Expandidos X Distância.

	Estimativa	Desvio Padrão	Valor t	$P(>\|t\|)$
Intercepto	5.270e+00	4.336e-02	121.55	<2e-16
Algoritmo A* (Heurística DNN)	1.213e-03	2.528e-05	48.00	<2e-16
Algoritmo enganoso, estratégia ds1	7,098e-01	5,106e-02	13,90	<2e-16
Algoritmo enganoso, estratégia ds2	1,670e+00	3,440e-02	48,55	<2e-16
Algoritmo enganoso, estratégia ds3	1,699e+00	3,445e-02	49,32	<2e-16
Algoritmo enganoso, estratégia ds4	6,441e+00	3,509e-02	183,58	<2e-16

Fonte: Próprio autor

Tabela 4.18 – Porcentagem de diferença de tempo de execução e nodos expandidos em relação ao algoritmo A* (DNN Parcial).

	Tempo de execução	Nodos Expandidos
Algoritmo enganoso, estratégia ds1	+54.27%	+103.36%
Algoritmo enganoso, estratégia ds2	+101.27%	+431.22%
Algoritmo enganoso, estratégia ds3	+98.54%	+446.85%
Algoritmo enganoso, estratégia ds4	+12124.17%	+62603.35%

Fonte: Próprio autor

Comparando as 3 versões dos algoritmos de busca executados no Mapa 1 (Figuras 4.1, 4.7, e 4.9), é possível perceber que a redução de tempo de execução atingida com a aplicação parcial da DNN nos algoritmos de busca não é tão alta quanto a aplicação completa da DNN nestes algoritmos (dados apresentados na sessão 4.2). Porém, ainda apresenta um ganho de desempenho considerável, onde o algoritmo enganoso com estratégia *ds4* teve um tempo de execução médio 123,24% mais rápido do que a execução deste algoritmo sem o uso de DNN, e esse ganho fica ainda maior quando caminhos de longa distância são considerados. Os nodos explorados pelos algoritmos de busca também apresentaram uma redução similar, onde é possível observar uma relação quase que direta entre as métricas de avaliação da Tabela 4.18. A seguir, na Tabela 4.19, são apresentados os resultados dos testes de diferença de custo desta aplicação. Estes resultados são comparados com os resultados dos algoritmos de busca calculados sem o uso da DNN (mesmo método utilizado na sessão anterior deste trabalho).

Tabela 4.19 – Média da diferença de custo entre os caminhos retornados pelos algoritmos de busca sem o uso de DNN e os algoritmos de busca com DNN parcial.

	Mapa 1	Mapa 2	Mapa 3
Algoritmo A* (Heurística DNN)	0,76%	0,69%	1,47%
Algoritmo enganoso, estratégia ds1	1,04%	1,31%	1,75%
Algoritmo enganoso, estratégia ds2	8,95%	7,77%	9,91%
Algoritmo enganoso, estratégia ds3	8,59%	7,62%	9,93%
Algoritmo enganoso, estratégia ds4	9,76%	8,38%	10,72%

Fonte: Próprio autor

A diferença de custo nos algoritmo *A** e o algoritmo enganoso com estratégia *ds1* é igual nas duas aplicações de DNN. Isso deve-se ao fato de elas não fazerem uso da predição de custo ótimo (pois esses algoritmos não fazem verificação de engano em nenhum momento). Porém, o algoritmo enganoso com as estratégias *ds2*, *ds3* e *ds4* obtiveram um grande ganho de precisão associado ao custo dos caminhos quando comparados com a aplicação completa da DNN nos algoritmos de busca (Tabela 4.14). Na Tabela 4.20, é apresentada uma análise de engano da aplicação parcial da DNN nos algoritmos testados. Esta análise é similar aquela apresentada na sessão 4.2.

Mesmo que a aplicação parcial da DNN nos algoritmos de busca apresentou custos mais

Tabela 4.20 – Média da diferença do número de passos reveladores de cada estratégia de busca em relação ao valor mínimo possível (DNN Parcial).

	Mapa 1	Mapa 2	Mapa 3
Algoritmo A* (Heurística DNN)	+68,19%	+81,27%	+75,05%
Algoritmo enganoso, estratégia ds1	+18,99%	+19,57%	+15,77%
Algoritmo enganoso, estratégia ds2	+59,85%	+64,08%	+59,57%
Algoritmo enganoso, estratégia ds3	+50,45%	+57,23%	+59,37%
Algoritmo enganoso, estratégia ds4	+29,74%	+32,27%	+23,81%

Fonte: Próprio autor

próximos aos testes destes algoritmos sem o emprego de DNN, os valores de engano encontrados são piores do que aqueles calculados na aplicação completa da DNN nos algoritmos de busca testados. Mesmo com esta perda de capacidade de engano, além de apresentar um ganho de desempenho inferior a aplicação completa da DNN nos algoritmos de busca, a aplicação parcial da DNN ainda têm um nicho prático real. Muitas vezes estas diferenças de engano não vão ser perceptíveis para um observador humano, caso seja necessário calcular caminhos mais rapidamente do que usar os algoritmos enganosos sem o uso de DNN. Com menos desvios de trajeto do que aqueles retornados pela aplicação completa da DNN nos algoritmos de busca, a aplicação parcial é uma boa opção para resolver muitos problemas de aplicação.

5 CONCLUSÃO

A computação de caminhos enganosos com o apoio de DNN é um assunto de pesquisa relevante para diferentes aplicações de IA. Embora muito ainda possa ser explorado nesse contexto, este trabalho aborda dois problemas distintos presentes na área. Primeiro, a computação de caminhos enganosos em terrenos com características topográficas, um contexto previamente não explorado para computação de caminhos enganosos. Segundo, o problema de desempenho de computações de algoritmos de busca de caminhos enganosos, onde o uso de DNN para otimizar algoritmos que computam estes caminhos pode ser investigado.

Os caminhos enganosos gerados pelas quatro estratégias enganosas testadas neste trabalho apresentaram comportamentos satisfatórios, com métricas de execução em linha com aquelas apresentadas em Masters e Sebastian (2017). Mais ainda, os caminhos computados apresentaram características visualmente realistas, as quais consideraram a variação do relevo no trajeto.

Os resultados obtidos com a aplicação de DNN como forma de otimização dos algoritmos de busca de caminhos testados foram mistos. Por um lado, os resultados apresentados na Sessão 4.2 demonstram um grande potencial de redução de tempo de execução e nodos expandidos. Porém, devido a natureza dos algoritmos que computam as estratégias enganosas, a forma que a DNN foi aplicada neste trabalho resultou em desvios significativos dos caminhos comparados com aqueles calculados sem DNN. Isso resultou em uma perda da capacidade enganosa associada aos caminhos computados. Devido a essas perdas, testes com uma aplicação parcial da DNN no algoritmo de busca foram realizados, onde a DNN é apenas utilizada para otimizar as execuções do algoritmo A*. Isso resultou em um custo de caminho médio menor e mais próximo da versão dos algoritmos de busca que não utilizaram a DNN. Contudo, o ganho de desempenho não foi tão grande em relação a aplicação completa da DNN, e os caminhos apresentaram um maior número de passos reveladores do que nas outras alternativas.

Existe várias formas que este trabalho pode ser expandido no futuro. Por exemplo, é possível explorar as técnicas analisadas em variações mais complexas de algoritmos de busca de caminhos enganosos tal como apresentados em Xu et al. (2020b). Neste trabalho, também se considerou visibilidade completa para o observador. Porém, em contextos reais é mais provável trabalhar com campos de visão limitados. A aplicação de aprendizado de máquina deste trabalho foi efetiva em reduzir o tempo de processamento dos algoritmos. Porém, apresentou um aumento de custo e redução de engano significativo, deixando um espaço para um estudo mais específico sobre outras formas aplicar DNN neste contexto.

REFERÊNCIAS BIBLIOGRÁFICAS

ABADI, M. et al. **TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems**. 2015. Software available from tensorflow.org. Disponível em: <https://www.tensorflow.org/>.

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. 1–11, 04 2015.

ALOM, M. Z. et al. A state-of-the-art survey on deep learning theory and architectures. **Electronics**, v. 8, n. 3, 2019. ISSN 2079-9292. Disponível em: <https://www.mdpi.com/2079-9292/8/3/292>.

BOTEA, A. et al. Pathfinding in games. In: ____. [S.l.: s.n.], 2013. p. 21–31. ISBN 978-3-939897-62-0.

BOTEA, A.; MüLLER, M.; SCHAEFFER, J. Near optimal hierarchical path-finding (hpa*). **Journal of Game Development**, v. 1, 01 2004.

CAI, Z. et al. Deceptive path planning in dynamic environment. In: **2020 3rd International Conference on Advanced Electronic Materials, Computers and Software Engineering (AEMCSE)**. [S.l.: s.n.], 2020. p. 203–207.

CHAGAS, C. et al. Hierarchical and smoothed topographic path planning for large-scale virtual simulation environments. **Expert Systems with Applications**, v. 189, p. 116061, 2022. ISSN 0957-4174. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0957417421014019>.

CHOI, S. et al. Global path planning on uneven elevation maps. In: **2012 9th International Conference on Ubiquitous Robots and Ambient Intelligence (URAI)**. [S.l.: s.n.], 2012. p. 49–54.

CHOLLET, F. et al. **Keras**. GitHub, 2015. Disponível em: <https://github.com/fchollet/keras>.

DANIEL, K. et al. Theta*: Any-angle path planning on grids. **CoRR**, abs/1401.3843, 2014. Disponível em: <http://arxiv.org/abs/1401.3843>.

FRAUNHOLZ, D. et al. Demystifying deception technology:a survey. 04 2018.

HART, P. E.; NILSSON, N. J.; RAPHAEL, B. A formal basis for the heuristic determination of minimum cost paths. **IEEE Transactions on Systems Science and Cybernetics**, v. 4, n. 2, p. 100–107, 1968.

HURWITZ, E.; MARWALA, T. Learning to bluff. In: **2007 IEEE International Conference on Systems, Man and Cybernetics**. [S.l.: s.n.], 2007. p. 1188–1193.

LI, G. et al. Ann: A heuristic search algorithm based on artificial neural networks. In: **Proceedings of the 2016 International Conference on Intelligent Information Processing**. New York, NY, USA: Association for Computing Machinery, 2016. (ICIIP '16). ISBN 9781450347990. Disponível em: <https://doi.org/10.1145/3028842.3028893>.

MACAL, C. Everything you need to know about agent-based modelling and simulation. **Journal of Simulation**, v. 10, p. 144–156, 05 2016.

MACAL, C. M. Everything you need to know about agent-based modelling and simulation. **Journal of Simulation**, Taylor Francis, v. 10, n. 2, p. 144–156, 2016. Disponível em: <https://doi.org/10.1057/jos.2016.7>.

MASTERS, P. Goal recognition and deception in path-planning. In: . [S.l.: s.n.], 2019.

MASTERS, P.; SARDINA, S. Cost-based goal recognition for path-planning. In: **Proceedings of the 16th Conference on Autonomous Agents and MultiAgent Systems**. Richland, SC: International Foundation for Autonomous Agents and Multiagent Systems, 2017. (AAMAS '17), p. 750758.

MASTERS, P.; SEBASTIAN, S. Deceptive path-planning. In: **Proceedings of the 26th International Joint Conference on Artificial Intelligence**. [S.l.]: AAAI Press, 2017. (IJCAI'17), p. 43684375. ISBN 9780999241103.

MCCULLAGH, P.; NELDER, J. **Generalized Linear Models**. Springer US, 1983. (Monographs on Statistics and Applied Probability). ISBN 9780412238505. Disponível em: <https://books.google.com.br/books?id=OUitAQAACAAJ>.

NEISSE, C. **Redes Neurais Profundas na Computação de Heurísticas Para Algoritmos de Busca de Caminhos em Mapas Virtuais Contendo Elevação e Inclinação**. 2021. 49 p. Monografia (Trabalho de Conclusão de Curso) — Universidade Federal de Santa Maria, 2021.

NEISSE, C. et al. Investigating deep neural networks as heuristic functions for path planning with topographic terrain characteristics in agent-based simulation. In: **2022 IEEE 31st International Symposium on Industrial Electronics (ISIE)**. [S.l.: s.n.], 2022. p. 174–181.

ROOT, P.; MOT, J. D.; FERON, E. Randomized path planning with deceptive strategies. In: **Proceedings of the 2005, American Control Conference, 2005.** [S.l.: s.n.], 2005. p. 1551–1556 vol. 3.

Sebastian Sardina, Peta Masters. **Deceptive Path Planning (DPP) System for p4**. [S.l.]: GitHub, 2021. <https://github.com/ssardina-planning/p4-simulator-gr>. Acessado em 19 nov 2021.

STURTEVANT, N. R. Benchmarks for grid-based pathfinding. **IEEE Transactions on Computational Intelligence and AI in Games**, v. 4, n. 2, p. 144–148, 2012.

XU, K. et al. Improving the scalability of the magnitude-based deceptive path-planning using subgoal graphs. **Entropy**, MDPI AG, v. 22, n. 2, p. 162, Jan 2020. ISSN 1099-4300. Disponível em: <http://dx.doi.org/10.3390/e22020162>.

____. Single real goal, magnitude-based deceptive path-planning. **Entropy**, MDPI AG, v. 22, n. 1, p. 88, Jan 2020. ISSN 1099-4300. Disponível em: <http://dx.doi.org/10.3390/e22010088>.