







Learning Heuristics for Topographic Path Planning in Agent-Based Simulations

Henrique L. Krever¹^a, Thiago R. S. Leão¹^b, Juliano M. Pasa¹^c, Edison P. de Freitas²^d,
Raul C. Nunes¹^e and Luis A. L. Silva¹^f

¹Graduate Program in Computer Science, Federal University of Santa Maria, Av. Roraima n° 1000,
97105-900, Santa Maria, Brazil

²Graduate Program in Computer Science, Federal University of Rio Grande do Sul,
CP 15064, 91501-970, Porto Alegre, Brazil

Keywords: Topographic Path Planning, Heuristic Learning, Deep Neural Networks, Agent-Based Simulations.

Abstract: Path planning algorithms with Deep Neural Networks (DNN) are fundamental to Agent-Based Modeling and Simulation (ABMS). Pathfinding algorithms use various heuristic functions while searching for a route with a low cost according to different criteria. When such algorithms are applied to compute agent routes in simulated terrain maps represented by large numbers of nodes and where topographic movement constraints are present, the problem is that traditional heuristic functions lose quality since they do not capture important characteristics for target simulation problems. To approach this issue, this work investigates the training of DNNs with large numbers of (i) *topographic path costs* and (ii) *correction factors* for standard Euclidean distance heuristic estimations. The aim is to use these DNNs as heuristic functions to guide the execution of different A^* -based topographic path planning algorithms in agent-based simulations. The work approaches the heuristic learning and computation of agent routes in topographic terrain maps of different natures. To assess the performance of the proposed techniques, experimental results with path planning algorithms and alternative topographic maps are analyzed according to statistical models.


1 INTRODUCTION


Deep Neural Networks (DNN) (Goodfellow et al., 2016) are crucial to the resolution of several path search problems of Agent-Based Modeling and Simulation (ABMS) (Macal, 2016). Various challenges related to the computation of path planning algorithms with different natures, the use of terrain map representation and indexing structures, the exploration of pre-processed terrain map and path information, the search for better heuristic functions, and the consideration of different types and numbers of agents are investigated in the path planning literature (Abd Alfoor et al., 2015). With the exploration of DNNs, these issues also have an important role in planning relief-aware movement behaviors for agents in simu-


lated virtual environments.


Heuristic path planning algorithms search for the path with the lowest cost according to different characteristics of the simulated application problem, where different types of heuristic functions are used to guide the search process. The key role of these functions is to estimate the distance from a current position n to a goal position g . In the A^* -based pathfinding algorithms, the cost of each node n is determined by the function $f(n) = g(n) + h(n)$, where $g(n)$ is the current path cost, $h(n)$ is the heuristic function to estimate the distance between the current and goal nodes.


The quality of a heuristic estimation is related to the form in that the function approaches the actual distance to the goal, directly influencing the choice of terrain map nodes analyzed during the path search. Traditional heuristic functions (e.g., Euclidean distance) adequately guide the search for agent routes in many simulated maps that do not have many obstacles or other terrain movement constraints. The problem is that standard heuristic functions become ineffective when the pathfinding algorithms have to com-


^a <https://orcid.org/0000-0002-5791-5557>

^b <https://orcid.org/0000-0002-6137-2751>

^c <https://orcid.org/0000-0003-0319-7817>

^d <https://orcid.org/0000-0003-4655-8889>

^e <https://orcid.org/0000-0003-3228-4071>

^f <https://orcid.org/0000-0002-6025-5270>

pute realist routes for agents in terrains with varied topographic characteristics. Moreover, many complex search problems can rely on heuristics that compute suboptimal solutions for many reasons (e.g., (Spies et al., 2019), where computing better quality agent routes has a fundamental role in maintaining the realism and fluency of the many virtual simulations.

Recent works (Takahashi et al., 2019) (Jindal et al., 2017) (Wang et al., 2019) (Ariki and Narihira, 2019) (Li et al., 2016) (Kirilenko et al., 2022) (Neisse et al., 2022) (Weber et al., 2022) investigate the exploration of DNNs as heuristic functions for the resolution of path planning problems in different applications. Although relevant proposals have been presented, exploring DNN in pathfinding for ABMS applications is not a mature research field. To approach this issue, this work investigates the computation of paths in terrain maps with different relief characteristics (Ganganath et al., 2014) (Chen et al., 2009) (Chagas et al., 2022), permitting the evaluation of the effectiveness of alternative path planning approaches based on the DNN-based learning of heuristic functions.

The pathfinding algorithm's heuristic functions guide the map node analysis during the path search in simulated topographic terrain maps. Therefore, the more the DNN-based heuristic function approximates the shortest distance between two terrain map nodes, the fewer nodes need to be analyzed by the pathfinding algorithm. Most importantly, these functions can learn the topographic characteristics of a set of simulated terrain maps to better estimate relief-aware distances between start and destination positions. This work details a DNN architecture and how to prepare the path information to train it so that the DNN learns the heuristic function to be used by topographic path planning algorithms. Experimental results presented in this work are computed using two distinct A^* -based pathfinding algorithms, where alternative forms of training and using the DNNs in the heuristic computations are investigated. This work assesses the proposed approach in terms of the number of expanded nodes, execution time, and path cost (distance). The various pathfinding results computed in different terrain maps are also statistically analyzed.

The work is organized as follows: Section II starts reviewing pathfinding techniques. Then it discusses approaches for topographic path planning and the exploration of DNNs as heuristic functions for pathfinding; Section III details the DNN-based topographic pathfinding approaches investigated in this work; Section IV details how the proposed techniques were experimentally analyzed. Then it discusses the obtained testing results; Section V presents final remarks and

directions for future work.

2 BACKGROUND AND RELATED WORK

Path planning algorithms (Abd Alfoor et al., 2015) are based on alternative criteria to find a route for agents to move in a virtual terrain map. For this, a movement cost value should be considered from the different topographic characteristics of a terrain map. These cost computations can involve characteristics such as path distance, travel time, agent energy required for the movement, travel limitations related to the agent's physical capacities, logistic agent issues, and many other application-oriented factors. Although this work focuses on the computation of paths for agents to better move through the terrain relief, it actually aims to investigate how to learn characteristics like these in DNNs so that they are not lost/underused in heuristic path search computations.

Path search algorithms can work with or without heuristic functions composing the travel cost computations. A heuristic estimates the cost between any node and a destination on the map representation structure, being zero when applied to the destination. This estimate assists the pathfinding algorithm in choosing the next node to be analyzed during the search, indicating the most promising map node in relation to finding a path that leads to the destination.

Dijkstra's algorithm (Frana and Misa, 2010) does not use such a heuristic function. However, it is usually a first choice in developing many simulation systems because it has a straightforward implementation, good performance level, and calculation of the minimum path cost between nodes representing the virtual terrain map. The A^* algorithm (Hart et al., 1968) and others derived from it differ from Dijkstra in using a heuristic function for prioritizing the choice of nodes that should be better than others during the search. With this, the A^* -family of algorithms significantly reduces the processing time of the path search.

The Bi-Directional A^* algorithm (BiA^*) (Pohl, 1971) is a version of graph search algorithms that, using a heuristic, does the search from the start node towards the destination node, while simultaneously searching from the destination node towards the start node. The path is returned when one of these searches finds a node opened by the other search. Because of that, the algorithm sometimes ends up finding suboptimal paths, as it can finish the execution without having examined less costly nodes. That is not a problem for many simulation applications since these suboptimal paths are much more efficiently computed, where

memory and processing time constraints have to be considered in many agent-based simulations.

2.1 Related Works

First, this work approaches the computing of agent routes in terrain maps with topographic characteristics.

The authors of (Ganganath et al., 2014) formulated a heuristic for movements with lower energy costs for mobile robots on terrain with elevation, using an A^* -based algorithm. A graph was used to abstract a *Digital Elevation Model* (DEM) of a small area of 1 km^2 of a region of canyons. Each graph vertex represents a terrain point and has 8 edges for its neighbors. The terrain points have three coordinates, two for the position and one for the elevation. The tangent arc of the inclination between coordinates gives the inclination angle between two terrain points. This angle is used in formulating the energy cost model for traversing graph vertices and the algorithm's heuristic.

The work in (Chen et al., 2009) presents a path-planning algorithm for 3D game scenes. The scene is represented by a graph, where the vertices represent the polygons and the edges represent the polygons' adjacencies used in the scene rendering. The heuristic used by the search algorithm uses the Euclidean distance between nodes of the terrain representation structure. The terrain inclination information, represented by normal vectors, is used in the function (o, n) , where a linear combination of these vectors represents the difficulty of moving between the nodes o and n . A graph is used to abstract a DEM, where the graph vertices store the height information of the pixel it represents. The difficulty of local movement between neighboring vertices, expressed in the function g of the $f(n) = g(n) + h(n)$ cost computations, uses the terrain height difference, where the inclination between neighboring vertices is employed when calculating the distance between two adjacent nodes. This distance is computed using the Euclidean distance in the \mathbb{R}^3 .

The work in (Chagas et al., 2022) describes a pathfinding algorithm that produces smoothed paths where agents have terrain inclination constraints that they can cross during their simulated movement tasks. A hierarchical structure models the maps to represent large terrains, where the leaf nodes have normal vectors that represent terrain inclinations in the 4 adjacent directions. The angle of normal vectors is used to compute the local cost function $g(n)$ (i.e., the cost to traverse adjacent positions according to the terrain relief) of the proposed pathfinding algorithm. In this

way, the higher terrain inclinations are more costly to be traversed by the simulated agents.

Second, this work also investigates the exploration of DNNs in path search computations.

The authors of (Takahashi et al., 2019) address the computation of path search heuristic values with neural networks. A Convolutional Neural Network (CNN) extracts information from map images collected for heuristics learning. Having the state-space map and a three-dimensional matrix containing information from the goal node and the path orientation angle, the CNN produces a heuristic value using the map dimensions and the number of goal node orientations. The work also used the Dijkstra algorithm output to produce the training dataset. Then the A^* algorithm was used to evaluate the CNN results.

The neural network model detailed in (Jindal et al., 2017) predicts the travel distance between source and destination GPS coordinates. This algorithm combines this prediction with the time of the day to better estimate a taxi travel time. The proposed approach uses only raw GPS origin, destination coordinates, and time of day information to perform distance and time predictions.

The NASR (Neuralized A-Star) model described in (Wang et al., 2019) proposes using DNNs to learn the cost functions of a heuristic algorithm used by a custom route recommendation problem. Route suggestions are issued from path planning-oriented queries between source and destination positions. The model consists of a Recurrent Neural Network (RNN) to model the cost from the source position to the current position and a value network (functions that assign weights to nodes) to estimate the cost of the candidate position to the target position.

A CNN-based technique for learning heuristic functions for path planners is presented in (Ariki and Narihira, 2019). The network input is a map of obstacles and a destination position. The network output is a heuristic map, an image containing all heuristic values from the destination point. The heuristic map predicted by the CNN is used as a query table to provide a heuristic value during path planning.

The ANN^* presented in (Li et al., 2016) relies on a heuristic algorithm similar to the A^* , although it employs a modified heuristic function. This heuristic function is generated from a regression neural network, which is used to learn the map characteristics and to predict the difficulty of pathfinding. The algorithm uses the same cost function as the A^* algorithm, using a traditional heuristic function and a multiplier, aiming to predict the route complexity. This multiplier function is the result of the neural network, $f(n) = g(n) + hT(n) * hNN(n)$.

In (Kirilenko et al., 2022), different ways of training DNN to aid the pathfinding heuristic estimation are compared with the use of uniform cost terrain maps. Using pre-computed path information, one performs DNN training to predict the absolute path cost (*cost-to-go*) and the other to predict a *correction factor* for a Euclidean distance heuristic estimation. Experimental results show that the second approach is superior because it finds more optimal cost paths. The results also demonstrate that, in some cases, A^* with DNN to predict the *cost-to-go* opened more nodes than the standard A^* algorithm. This did not occur with the DNN trained to return a *correction factor* to be used by the weighted A^* algorithm.

In (Neisse et al., 2022), two DNN approaches to assist in the A^* -based path planning over a set of topographic terrain maps are explored: (1) $h_{sub}(n) = DNN(n, m)$, where a DNN is trained for each used 100x100 pixel map representing terrain relief characteristics, making the heuristic predictions $h_{sub}(n)$ for that map only, and (2) $h_{subAll}(n) = DNN(n, m)$, where a single DNN is trained to make heuristic prediction $h_{subAll}(n)$ for three different topographic maps, where a map identifier is used as input in the DNN executions. Experiments showed that the second approach is more promising because it presents a good pathfinding performance, guaranteeing a smaller expansion of nodes and, consequently, execution time.

DNNs are explored in (Weber et al., 2022) to support computing A^* -based safe paths in topographic terrain maps: paths that deviate as much as possible from the field of view of observers at different map positions. Two different DNN strategies are compared: (1) $h(n) = DNN_1(n)$, where the DNN is trained to predict an absolute path cost in the \mathbb{R}^3 , while other path cost components are considered during the path search runtime only, and (2) $h(n) = DNN_2(n)$, where the DNN is trained to predict topographic path costs that include a path safety cost component. Experimental results show that DNN approaches can open fewer nodes and have lower computing time than the topographic-aware A^* algorithm. Using the safety factor in the DNN2 training showed that other path cost components can be successfully inserted in the learning and, consequently, in the heuristic predictions used to support the path search.

3 DNNS AS HEURISTIC FUNCTIONS FOR COMPUTING PATHS IN TOPOGRAPHIC TERRAIN MAPS

Agent navigation behaviors for virtual simulation systems are most often modeled and implemented to reflect the actual physical behaviors of real-world agents. When simulations are performed in mountainous terrains, the agents' movement behaviors can be compromised because the relief roughness may cause all sorts of simulation problems. In many situations, the wheels of the simulated vehicles slip in the steep terrain slopes, these slopes make the vehicles capsize in the simulations, and the agents can simply try to use routes that make the movement progress almost impossible. The computation of paths considering these risks is paramount for the proper functioning and integrity of the agent-based simulations. As investigated in this work, path computations should prioritize determining routes that consider the various characteristics of the simulated terrain topographies.

To implement virtual simulation systems in which the terrain maps represent characteristics of real-world mountainous regions, this work models the virtual terrain as a regular grid with non-uniform costs. This representation is based on a DEM of the topographic terrain, where map nodes represent each pixel of the terrain elevation model. Elevation information is calculated similarly to in (Chagas et al., 2022). In it, the Euclidean distance function in the \mathbb{R}^3 is used to compute the $g(n)$ and $h(n)$ values of the $f(n) = g(n) + h(n)$ cost function used in the path search. Then, the topographic terrain information is converted to cost values stored in the map mode representation structure. This cost is used to make up the cost of crossing two adjacent nodes of the terrain representation map and the training of DNNs for estimating the topographic route costs between two terrain positions.

Figure 1 describes the calculation of the topographic path cost between two adjacent terrain nodes A and B . Each map node has a resolution of 30 meters in the real world. The height of each node (i.e., $Height(A)$ and $Height(B)$) varies according to the relief elevation in that terrain position. The height difference between two adjacent map nodes $|Height(B - A)|$ and the map node resolution $Diff(x, y)$ is used to calculate the topographic distance $Cost(A, B)$ given by the triangle's hypotenuse. Then the resulting value is used as a local cost value between adjacent map nodes.

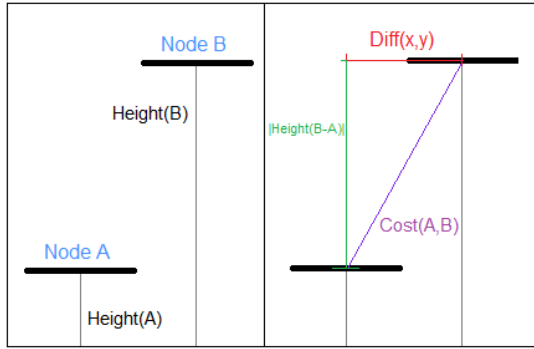


Figure 1: Cost calculation based on the height difference between adjacent nodes in the terrain map representation.

3.1 Heuristic Topographic Path Cost Computations with DNNs

This work investigates DNNs as heuristic functions for the A^* and BiA^* pathfinding algorithms (Abd Alfoor et al., 2015) when computing paths in terrains with topography. To assess the impact of alternative forms of DNN training with pre-computed path costs between start S and goal G terrain map nodes, the $f(n) = g(n) + h(n)$ cost function used by these algorithms uses the following heuristic formulations:

- $h(n) = DNN_{absCost}(n)$, where the $DNN_{absCost}(n)$ returns *absolute path cost* heuristic values;
- $h(n) = DNN_{errorCoeff}(n) * h'(n)$, where the $DNN_{errorCoeff}(n)$ returns an *error coefficient* value to be used with the $h'(n)$, which computes Euclidean distance heuristic values in the \mathbb{R}^3 ;

$DNN_{absCost}$ - the computation of paths on terrains with topography is motivated by the works presented in (Neisse et al., 2022) and (Weber et al., 2022). In them, the heuristic function $h(n)$ of the used pathfinding algorithm is fully replaced by the $DNN(n)$, such that $h(n) = DNN(n)$. The $DNN_{absCost}$ is trained with *absolute path cost* values. These path costs are pre-computed between start S and goal G terrain map nodes.

$DNN_{errorCoeff}$ - the DNN is trained to compute an *error coefficient* value. This error is related to the heuristic distance values computed using the Euclidean distance function in the \mathbb{R}^3 . Motivated by the work presented in (Kirilenko et al., 2022), the $DNN_{errorCoeff}$ is trained with pre-computed error coefficients, where the resulting DNN returns a correction rate to be applied to the path costs returned when standard (map-independent) heuristic distance functions are used. The DNN is actually trained with

the topographic path costs $dist_{Topog}(n)$ divided by cost estimates for the same paths returned by the Euclidean distance function $dist_{Eucl}(n)$. Therefore, the values returned by the trained DNN are $DNN(n) = (dist_{Topog}(n) / dist_{Eucl}(n))$. In the runtime pathfinding computations, the values returned by the DNN are multiplied by those returned by the Euclidean distance function in the \mathbb{R}^3 between the considered start S and goal G map nodes. It means that the heuristic component $h(n)$ of the path cost calculations of $f(n)$ becomes $dist_{Eucl}(n) * DNN_{errorCoeff}(n)$. Therefore, the path search cost function takes the form of a weighted cost function $f(n) = g(n) + w * h(n)$ (Ebendt and Drechsler, 2009).

The DNN architecture (the number of neurons and layers) used in this work was determined experimentally. A fully connected Feed-Forward DNN model is used, with the hidden layers having certain symmetry in relation to the number of neurons per layer. From the performed tests, the model that obtained the most promising results was similar to the DNN models described in (Neisse et al., 2022) and (Weber et al., 2022). In this model, the number of neurons is expanded, then contracted in the hidden layers. The DNN architecture is the following: a) the linear function for the input layer, with 7 neurons (i.e., a terrain map ID and the (x, y, z) coordinates of start and goal map positions), b) the ReLU function for the three hidden layers, with [400, 500, 400] neurons, and c) the Sigmoid function for the output layer with 1 neuron to return either the *absolute path cost* value or the *correction factor* value.

3.2 The Computation of Heuristic Maps from DNN Estimates

The trained DNN calculates the heuristic function for the targeted pathfinding algorithms. It computes the heuristic value for the cost function in each node analysis during the search. In doing so, the execution of the pathfinding algorithms may not be as fast as required as the number of neuron calculations in the overall DNN architecture grows. That is because heuristic calculations using standard distance functions are much faster than those developed using the DNN elements. The problem is that traditional heuristic functions fail (e.g., due to the terrain topography, the heuristic results are too different from the actual path costs between the considered terrain positions) when other application-oriented components must be considered in the path calculations. To approach this problem, this work explores GPU parallelism functionalities to compute DNN-based heuristic function values during the pathfinding execution.

This approach involves the computation and use of a “heuristic map”.

A two-dimensional matrix can represent a heuristic map in many terrain maps. The matrix values record the topographic distances from all map positions S to a goal position G . This matrix is fed with the DNN distance predictions generated via the GPU parallel computations. This heuristic map is computed through the DNN execution and filled out once at the first step of each pathfinding execution. It means that the pathfinding algorithm retrieves the estimated path cost value in the previously computed heuristic map when it requires a heuristic value for a current node during the path search.

According to tests in this work, the delay time required to generate the heuristic map for all 300x300 topographic map instances is at most 600 ms. Moreover, this additional computing time required to generate the heuristic map starts to pay off when the routes’ complexity in the used terrain map increases. For large-scale and uneven terrain maps, where a large number of heuristic computations may have to be developed for the opening of a large number of nodes by the pathfinding algorithm, the GPU parallel computation of all heuristic map values in a single run is faster than the individual DNN computations of heuristic values when required.

4 EXPERIMENTS AND RESULTS

The experiments developed in this work aimed to evaluate the pathfinding techniques resulting from the two different DNN models as heuristic components for path search computations. Eight maps with different topographies were used in the tests. For each map, pairs of S/G map positions were generated, where the goal was to find the shortest topographic path between these positions. The A^* and BiA^* pathfinding algorithms were individually executed using the tested DNN-based heuristic functions - $DNN_{absCost}$ and $DNN_{errorCoeff}$ - for each pair of map positions. The heuristic function values returned by the DNNs were recorded in the generated heuristic maps. Whenever this map was required, it was computed and used during the execution of the tested pathfinding algorithms. For each algorithm execution, the resulting path’s topographic length (distance), the number of expanded nodes, and the execution time were measured and stored for statistical analysis.

In the terrain maps’ pre-processing activities, the following attributes were used to generate the dataset for the DNN training:

- *mapID*: representing the map used in the path

computations;

- $S(x, y, z)$: representing the start map position S ;
- $G(x, y, z)$: representing the goal map position G ;
- Either (i) the shortest distance $d_{Topog}(S, G)$ between the S and G positions, representing the *absolute path cost*, or (ii) the heuristic *correction factor* computed by $d_{Topog}(S, G) / d_{Eucl}(S, G)$, where $d_{Topog}(S, G)$ is the shortest topographic path cost and $d_{Eucl}(S, G)$ is the Euclidean distance in the \mathbb{R}^3 .

A sample of map positions equivalent to 10% of each terrain map was selected to generate the training dataset. The shortest topographic distance between these positions was calculated using a parallel version of the topographic-aware Dijkstra algorithm (Harish and Narayanan, 2007). Moreover, the Dijkstra execution was adapted not to have a target node, expanding to all map nodes. This way, the Minimum Spanning Tree (MST) was generated for the selected map nodes. Thus, it was possible to obtain the shortest topographic distance of any node n in relation to all other nodes accessible from it on the terrain map. Subsequently, the generated dataset was randomly shuffled, grouped, and separated into three different datasets. Then we used the cross-validation method during the training, where the 70/15/15 distribution was selected (70% for training, 15% for testing, and 15% for validation).

The computed distances between the map’s different S and G positions were stored in the CSV format. All computed distances are positive real values.

Table 1: Real-world topographic terrain maps and DNN training information.

Feature	Value
Different maps	8 maps (Figure 2)
Size of each map	300 x 300 pixels
Pixel resolution	30 m
Dimension of each map	81 km ²
Sample of each map	9,000 (10%)
Dataset size (computed paths)	$8 * \binom{9,000}{2} = 323,964,000$
Dataset memory size	16.5 GB
Dataset generation time	~48 hours
Train/test/validation sets	70/15/15

The $DNN_{absCost}$ and $DNN_{errorCoeff}$ used by the topographic A^* computations were trained in two DNNs. The DNNs were trained to less than 8.5% of error, where the Mean Absolute Percentage Error expressed the training error (Myttenaere et al., 2016) given by the MAPE function (1).

$$MAPE = \frac{100}{n} \sum_{t=1}^n \left| \frac{y_t - \hat{y}_t}{y_t} \right| \quad (1)$$

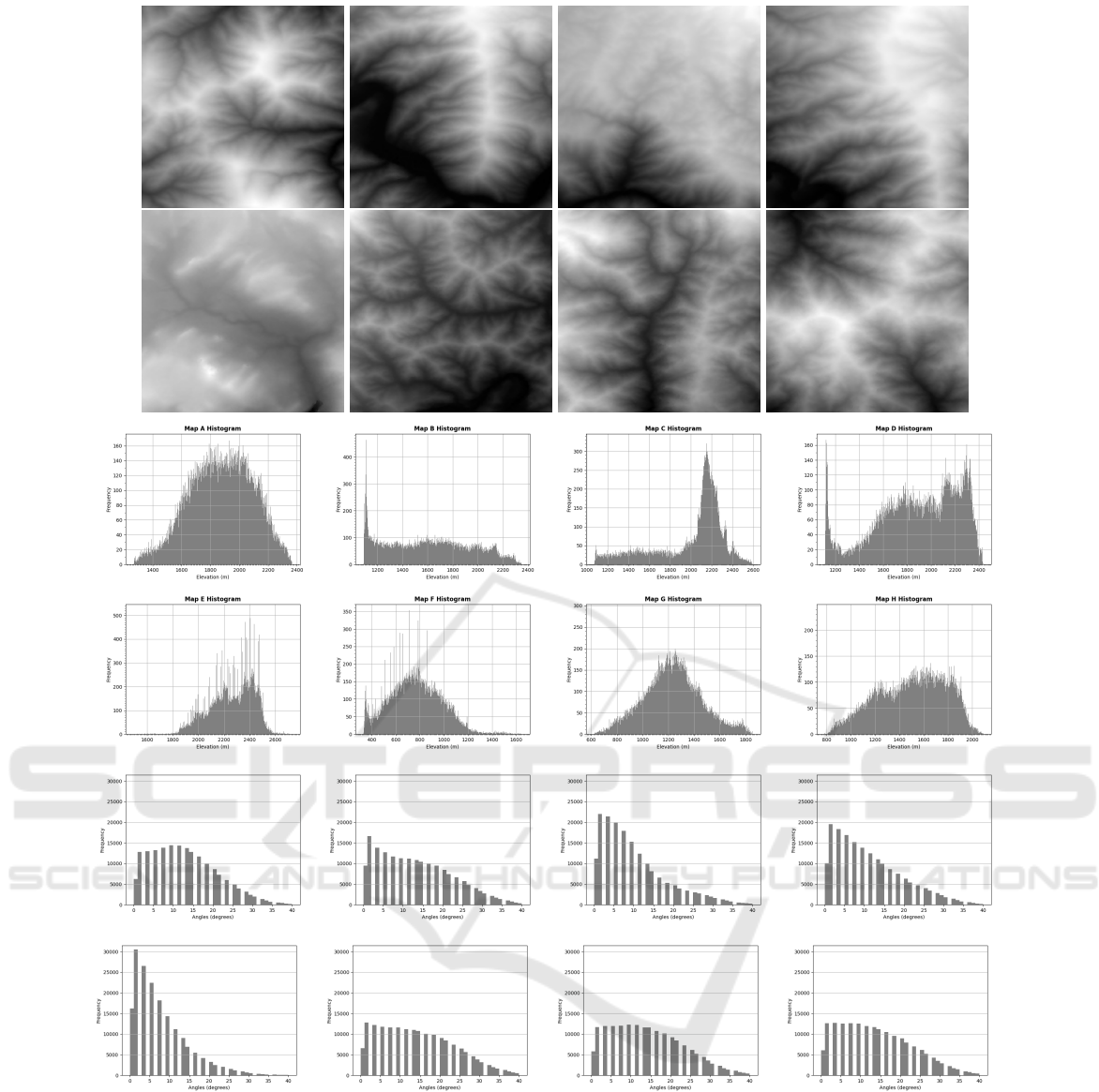


Figure 2: DEMs obtained from ((SRTM), 2013), elevation and inclination (between adjacent nodes) histograms for the real-world topographic terrain maps used in this work.

where $y_i - \hat{y}_i$ is the error between the predicted DNN value and the expected value, n is the cardinality of the dataset used to assess the trained DNN error.

In the DNN training, the two main parameters were the loss function and optimization. While the chosen loss function was the Mean Absolute Error, the optimization function was ADAM (Kingma and Ba, 2014). We explored the *minibatch* approach (Goodfellow et al., 2016) with a batch size of 32. A maximum number of 100 seasons was defined for the training duration. A premature stop criterion of 30 seasons passed without prediction improvements was also used. The DNNs were implemented and trained

in Python using the TensorFlow library (Abadi et al., 2016). Table 2 details the computing environment used throughout this work’s development.

Table 2: Computing environment used in this work.

CPU	Intel® Core™ i5-12400F
CPU cores	6 cores
CPU frequency	2.5 GHz / 4.4 GHz
RAM memory	(2x) 16 GB DDR5 2400MHz
GPU	GeForce GTX 3060
GPU cores	3,584
GPU memory	12 GB GDDR6
Operating system	Windows 11 Pro

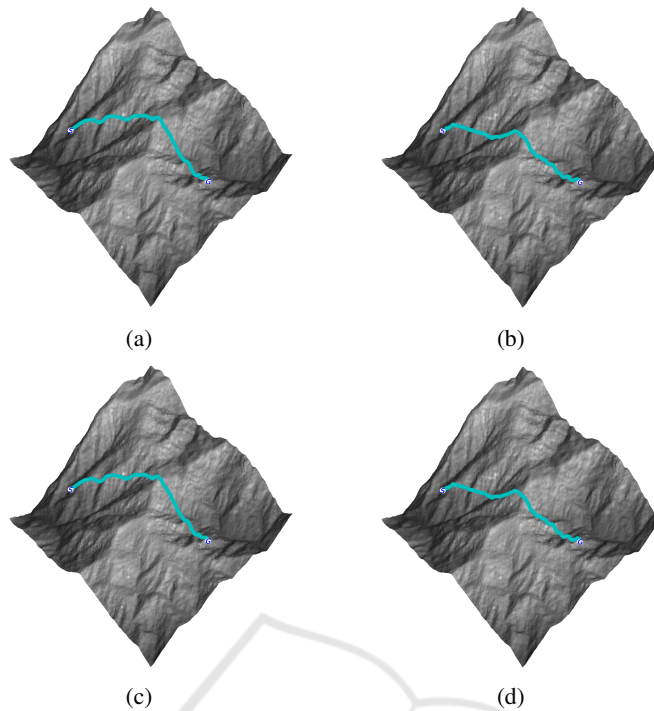


Figure 3: Paths computed with (a) $A^* DNN_{absCost}$, path cost = 7,788.06m; (b) $A^* DNN_{errorCoeff}$, path cost = 7,823.91m; (c) $BiA^* DNN_{absCost}$, path cost = 7,878.06m and (d) $BiA^* DNN_{errorCoeff}$, path cost = 7,817.65m.

A Generalized Linear Regression Model (GLRM) (McCullagh and Nelder, 1989) was used to statistically analyze the experimental pathfinding results. This model was computed using the R statistical software package (R Core Team, 2013). Details and examples of such kind of statistical analysis can be found in (Chagas et al., 2022) (Neisse et al., 2022) and (Weber et al., 2022).

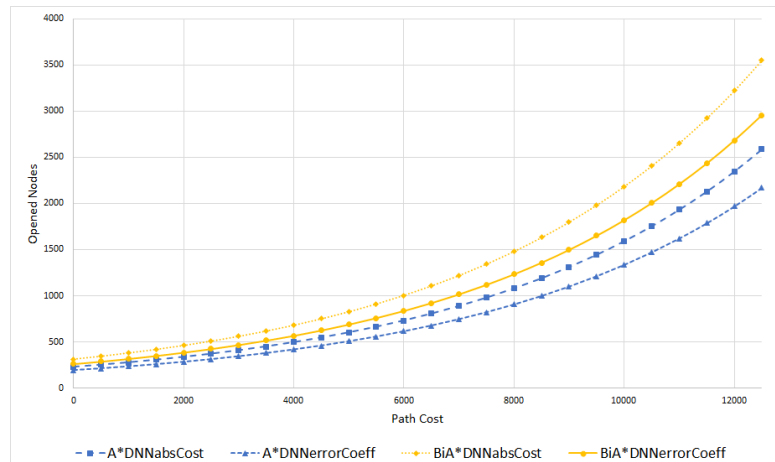
To compare the tested methods, the computed dataset consists of 6,350 paths in each map (50,800 in total) for each tested heuristic approach. For every path, the resulting topographic path cost (distance in meters), the execution time, and the number of opened nodes were recorded. To assess the effects of the tested DNN-based heuristic setups, the pathfinding results were detailed in terms of a) the expanded node values in relation to the topographic distance of the resulting path and b) the execution time values in relation to the distance of the resulting topographic path.

Instances of topographic paths between the same S and G positions computed by the tested methods in the same terrain map are presented in Figure 3. Despite the differences in the plotted topographic routes, they have quite similar costs. The statistical results obtained with the pathfinding tests are presented in Tables 3 and 4. All pathfinding results are statistically significant (Signif. level: 0.001, Std. Error: 0.01).

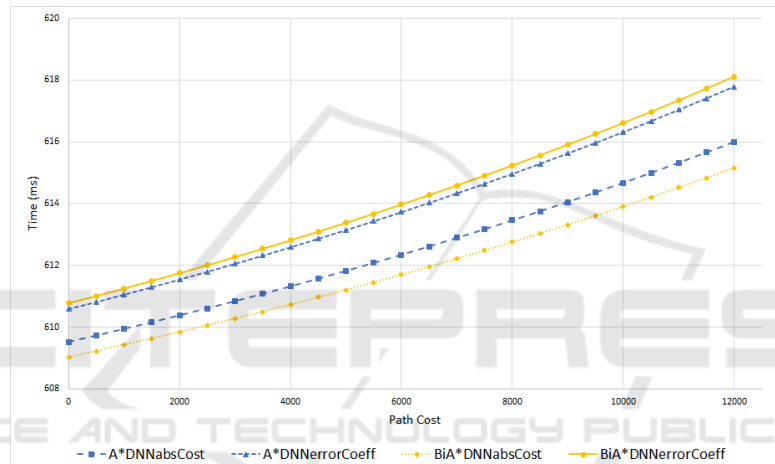
These (i) execution times and (ii) number of opened nodes results are also shown in Figure 4.

The test results show that the $DNN_{errorCoeff}$ opens a smaller number of nodes in relation to the $DNN_{absCost}$ with the A^* and BiA^* algorithms. With the trained DNNs, this result shows that the $DNN_{errorCoeff}$ better approximates the heuristic estimates of the actual topographic path costs. The A^* with $DNN_{errorCoeff}$ opens the lowest number of nodes among the tested techniques. It opens 16.07% fewer nodes than the A^* with the $DNN_{absCost}$, the base technique in this statistical analysis. Regarding the number of opened nodes, the A^* with the $DNN_{absCost}$ is second among the tested techniques. The BiA^* with the $DNN_{errorCoeff}$ opens 14.30% more nodes than the base method, and the BiA^* with the $DNN_{absCost}$ opens 37.33% more nodes than the base method. The fastest method is the BiA^* with the $DNN_{absCost}$. It runs 5.21% faster than the base method. Regarding computing time, the A^* with the $DNN_{absCost}$ is second among the tested techniques. The slowest method is the BiA^* with the $DNN_{errorCoeff}$, expending 13.20% more time than the base method. The A^* with $DNN_{errorCoeff}$ is 11.23% slower than the base method. The results indicate that the tested pathfinding algorithms with the $DNN_{errorCoeff}$ method is slower because they require the additional calculation of the Euclidean distances.

To assess the topographic path costs resulting



(a)



(b)

Figure 4: (a) Expanded nodes x topographic path cost, and (b) Execution time x topographic path cost.

from the tested pathfinding algorithms with the heuristics proposed in this work, the *Mean Absolute Percentage Error* (MAPE) function was used. Table 5 shows that the paths generated by DDN-based heuristics had a maximum cost difference of 1.34% in relation to the optimal path costs returned by the A^* with the Euclidean distance heuristic function.

The experiments indicate that the pathfinding method choice is based on a tradeoff between expanded nodes and execution time. While the $DNN_{errorCoeff}$ method presents fewer memory costs, it is slower than the $DNN_{absCost}$ method. In addition, the $DNN_{absCost}$ and $DNN_{errorCoeff}$ methods resulted in shorter topographic paths when used with the A^* algorithm in comparison to the topographic path costs of the BiA^* algorithm. Overall, the tested approaches generalized the eight different terrain topographies well, meaning that they can approach the training of new terrains.

In summary, the proposals and experimental results presented in this paper analyze the combination of the DNNs and pathfinding techniques for the heuristic map generation and use, the exploration of the parallel version of the topographic-aware Dijkstra algorithm (Harish and Narayanan, 2007) to construct the training dataset, and the DNN model to reproduce the heuristic computations in different topographic terrain maps. This work analyzes DNN heuristic methods for learning a large number of path cost instances (i.e., shortest topographic path distances) between two S/G terrain map positions and the consequent learning of targeted map topographic characteristics. This work also presents the statistical analysis of experimental pathfinding results, where the tested DNN-based algorithms' performance is analyzed.

Table 3: Statistical results for execution time x topographic path cost.

	Estimate	Std. Error	t Value	Pr(> t)
Intercept	-4.654	0.0059	-784.69	<2e-16
A^* $DNN_{absCost}$	0.0000	0.0000	70,586.76	<2e-16
A^* $DNN_{errorCoeff}$	0.1064	0.0075	14.18	<2e-16
BiA^* $DNN_{absCost}$	-0.0535	0.0074	-7.27	<3.57e-13
BiA^* $DNN_{errorCoeff}$	0.1240	0.0069	17.93	<2e-16

Table 4: Statistical results for expanded nodes x topographic path cost.

	Estimate	Std. Error	t Value	Pr(> t)
Intercept	5.4290	0.0038	1,424.73	<2e-16
A^* $DNN_{absCost}$	0.0002	0.0000	418.82	<2e-16
A^* $DNN_{errorCoeff}$	-0.1752	0.0033	-52.39	<2e-16
BiA^* $DNN_{absCost}$	0.3172	0.0033	96.68	<2e-16
BiA^* $DNN_{errorCoeff}$	0.1337	0.0031	43.37	<2e-16

Table 5: Percentage of DDN-based path costs compared to the optimal path costs (computed with the A^* algorithm with Euclidean distance heuristic function).

	$DNN_{absCost}$	$DNN_{errorCoeff}$
A^*	1.31%	0.87%
BiA^*	1.34%	0.92%

5 FINAL REMARKS

The computation of path routes for agents in topographic terrain maps is a relevant problem for many agent-based simulation applications. In this context, this work shows that the heuristic functions learned by DNNs can support the path search in topographic map representations. The use of the DNN allows the implementation of the search for paths which combines path pre-processed information – the pre-computed topographic distances in the terrain map that are later learned into DNN memory structure – with the actual path cost computations developed during the execution time of different path planning algorithms.

Future work in our project will aim at the hierarchical and irregular capture of the extremely large real-world topographic terrain maps, including a large number of maps with more distinct topographic characteristics (e.g., varying from flat areas to mountain regions), in addition to other required adjustments for DNN-based pathfinding algorithms. Moreover, we aim to investigate the learning of characteristics that could be more independent of the used terrain maps, and the investigation of other DNN architectures, with special attention to the computation of topographic-aware agent routes for agent-based simulations.

ACKNOWLEDGMENT

We thank the Brazilian Army Strategic Program AS-TROS for financial support through the SIS-ASTROS GMF project (898347/2020) - TED 20-EME-003-00.

REFERENCES

- Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., Devin, M., Ghemawat, S., Irving, G., and Isard, M. (2016). Tensorflow: A system for large-scale machine learning.
- Abd Algfoor, Z., Sunar, M. S., and Kolivand, H. (2015). A comprehensive study on pathfinding techniques for robotics and video games. *International Journal of Computer Games Technology*, 2015.
- Ariki, Y. and Narihira, T. (2019). Fully convolutional search heuristic learning for rapid path planners. *arXiv preprint arXiv:1908.03343*.
- Chagas, C., Zacarias, E., de Lima Silva, L. A., and Pignaton de Freitas, E. (2022). Hierarchical and smoothed topographic path planning for large-scale virtual simulation environments. *Expert Systems with Applications*, 189:116061.
- Chen, S., Shi, G., and Liu, Y. (2009). Fast path searching in real time 3d game. In *WRI Global Congress on Intelligent Systems*, volume 3, pages 189–194.
- Ebendt, R. and Drechsler, R. (2009). Weighted A^* search – unifying view and application. *Artificial Intelligence*, 173(14):1310–1342.
- Frana, P. L. and Misa, T. J. (2010). An interview with edsgar w. dijkstra. *Communications of the ACM*, 53(8):41–47.
- Ganganath, N., Cheng, C.-T., and Tse, C. K. (2014). Finding energy-efficient paths on uneven terrains. In *2014 10th France-Japan/ 8th Europe-Asia Congress on Mechatronics (MECATRONICS2014- Tokyo)*, pages 383–388.

- Goodfellow, I., Bengio, Y., Courville, A., and Bengio, Y. (2016). *Deep learning*, volume 1. MIT press Cambridge.
- Harish, P. and Narayanan, P. J. (2007). Accelerating large graph algorithms on the gpu using cuda. In A., S., P., M., B., R., and P., V., editors, *High Performance Computing – HiPC 2007*, pages 197–208. Springer.
- Hart, P. E., Nilsson, N. J., and Raphael, B. (1968). A formal basis for the heuristic determination of minimum cost paths. *IEEE Transactions on Systems Science and Cybernetics*, 4(2):100–107.
- Jindal, I., Qin, T., Chen, X., Nokleby, M., and Ye, J. (2017). A unified neural network approach for estimating travel time and distance for a taxi trip. *arXiv preprint arXiv:1710.04350*.
- Kingma, D. P. and Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.
- Kirilenko, D., Andreychuk, A., Panov, A., and Yakovlev, K. (2022). Transpath: Learning heuristics for grid-based pathfinding via transformers. *arXiv preprint arXiv:2212.11730*.
- Li, G., Wang, G., Wang, Q., Fei, F., Lü, S., and Guo, D. (2016). Ann: A heuristic search algorithm based on artificial neural networks. In *Proceedings of the 2016 International Conference on Intelligent Information Processing*, pages 1–9.
- Macal, C. (2016). Everything you need to know about agent-based modelling and simulation. *Journal of Simulation*, 10:144–156.
- McCullagh, P. and Nelder, J. A. (1989). *Generalized Linear Models*, volume 37. CRC Press.
- Myttenaere, A. d., Golden, B., Grand, B. L., and Rossic, F. (2016). Mean absolute percentage error for regression models. *Neurocomputing*, 192:38–48.
- Nesse, C., Soares, J. L., Silva, L. A., and Freitas, E. P. (2022). 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)*, pages 174–181. IEEE.
- Pohl, I. (1971). Bi-directional search. in machine. *Intelligence*, 6:124–140.
- R Core Team (2013). *R: A Language and Environment for Statistical Computing*. Vienna, Austria.
- Spies, M., Todescato, M., Becker, H., Kesper, P., Waniek, N., and Guo, M. (2019). Bounded suboptimal search with learned heuristics for multi-agent systems. *Proceedings of the AAAI Conference on Artificial Intelligence*, 33(01):2387–2394.
- (SRTM), N. S. R. T. M. (2013). Shuttle radar topography mission (srtm) global. Distributed by Open-Topography. Accessed: 2023-04-12. Available: <https://doi.org/10.5069/G9445JDF>.
- Takahashi, T., Sun, H., Tian, D., and Wang, Y. (2019). Learning heuristic functions for mobile robot path planning using deep neural networks. In *Proceedings of the International Conference on Automated Planning and Scheduling*, volume 29, pages 764–772.
- Wang, J., Wu, N., Zhao, W. X., Peng, F., and Lin, X. (2019). Empowering a* search algorithms with neural networks for personalized route recommendation.
- Weber, C. M., Freitas, E. P., and Silva, L. A. (2022). Safe and topographic path planning with deep neural networks. In *2022 21st Brazilian Symposium on Computer Games and Digital Entertainment (SBGames)*, pages 1–6. IEEE.