Investigating Deep Neural Networks as Heuristic Functions for Path Planning with Topographic Terrain Characteristics in Agent-Based Simulation

Claiton Neisse*, Juliano L. Soares*, Luis A. L. Silva*, Edison P. Freitas†
*Applied Computing Department, Federal University of Santa Maria - UFSM; Santa Maria, Brazil
{chneisse, jlsoares, luisalvaro}@inf.ufsm.br
†Institute of Informatics, Federal University of Rio Grande do Sul - UFRGS; Porto Alegre, Brazil
epfreitas@inf.ufrgs.br

***Abstract*—Agent-Based Modelling and Simulation (ABMS) is a fundamental paradigm to the investigation of important application problems. A way to expand this paradigm is to integrate Deep Learning and Path Planning techniques into the development of simulation systems. The problem is that the integration of these Artificial Intelligence (AI) approaches in the realistic and optimized implementation of agent behaviors is a not mature research subject yet, mainly with respect to the exploration of path planning algorithms that use topographic terrain information in the computation of routes for real-world simulated agents. With the pre-processing and representation of terrain relief characteristics, and latter use of trained DNNs from datasets representing pre-computed topographic-aware paths, this work investigates the path planning exploration of DNN-based heuristic functions. The aim is to assess how such heuristics can be used by the path planning algorithm to support time-limited computations of topographic-based path routes in simulated terrain environments.**

***Index Terms*—Topographic path planning, Heuristic path planning, Deep neural networks, Agent-based simulations**

I. Introduction

Agent-Based Modelling and Simulation (ABMS) [1] is a state-of-art paradigm to the investigation of real-world applications. Agent-based simulations are relevant to training and exercising, hypotheses tests, planning and many other usages, with reduced costs compared to execution of the targeted tasks in the real-world. A way to enhance agent-based simulations is to introduce Deep Learning into it so that the simulated agents can learn from real-world and/or simulated data to execute optimized and realistic computations in the simulated scenarios (e.g. [2]). As investigated in this work, such Deep Learning approach is able to optimize the agent computations regarding topographic path planning so that they do not impair the fluency and realism of the simulated problems.

Approaches of Deep Neural Networks (DNN) [3] and Path Planning [4] have been recently studied outside the ABMS context [5] [6] [7] [8]. The computation of topographic-aware paths have also been approached in different forms [9] [10] [11], although still not exploring Deep Learning. These path planning works explicitly consider the terrain relief in the determination of low cost routes for the targeted agents. Despite these works, in which different DNN models are investigated as tools to support the enhancement of path planning algorithms, the use of DNNs in the search for paths on terrain representations capturing topographic information is an under-investigated problem in ABMS.

In this work, the topology of the real-world terrains used in the implementation of agent-based simulation environments is pre-processed as part of simulation preparation tasks. Then the resulting data used for DNN training is explored during the runtime path planning required by the agents inserted into the simulations. In doing so, the contributions of this work are: i) the analysis of DNNs as heuristic functions to be used by realistic topographic-aware path planning algorithms targeting real-world terrain maps used by agent-based simulation systems, and ii) the statistical performance evaluation of DNN-based heuristics in the optimized search for paths in different terrain maps containing terrain inclination and height information.

II. Problem Statement

Path planning algorithms [4] used by agent-based simulation tools [12] rely on different kinds of heuristics in the composition of their path cost functions. Given the required simulation realism regarding the topographic movement of the agents, such heuristics ought to estimate the relief cost between any node of the terrain map and a goal position. This relief travel cost is related to the nature of the addressed application problem, where the cost function can model: the travel time considering the terrain relief, the energy required for traveling along the ups and downs of the resulting route, the topographic limitations related to the safe movement of the agents, among others. The problem is that multi-agent simulations need to have forms of computing such realistic topographic routes while dealing with reduced computational time limits regarding the path search. This issue has particular relevance for the SIS-ASTROS project [13]. This project involves the development and improvement of real-world agent-based simulations, where different agents need to consider the terrain topography while executing realistic movements in virtual terrain scenarios.

To support the movement of agents in the simulated environment, the path planning [4] addresses the minimization of the path costs. To do so, a heuristic estimates the cost between any node on the represented map and a destination. This estimation assists the algorithm in the selection of the next node to be analyzed during the search of a path, indicating the most promising node that considers the task of finding a route leading to the destination. In general, the cost function is expressed by (1), where g expresses the cost between the origin (o) and the node being evaluated (n), h expresses the heuristic cost between the node being evaluated (n) and the destination (d) and f indicates the total cost of the node (n).

$$f(n) = g(o, n) + h(n, d) \tag{1}$$

Distance metrics are often used in the h formulation of (1). Despite leading to optimal paths, such traditional heuristics disregard map obstructions and other movement constraints of the targeted agents. It means that terrain representation nodes can be expanded beyond what it is required during the search of a path. That is a significant problem for simulations that are run in large-scale real-world terrain maps. To consider topographic terrain characteristics in the heuristic h, such as terrain relief inclinations and heights, for instance, requires the exploration of Deep Learning. Aiming at the optimization of topographic path planning, the implementation of a DNN may be as effective as the use of hierarchical terrain search and representation strategies, such as the hierarchical search in QuadTrees used in the SIS-ASTROS simulation system [14] [15], but with simpler implementation and maintenance procedures compared to the complexity involved in the development of the hierarchical approaches. Despite this advantage, the problem is exactly to determine how to make use of the Deep Learning technique so that the topographic path planning can benefit from it, then resulting on the desired improvement of the ABMS paradigm. The goal of this work is to address this problem providing evidence that it is achievable.

III. Related Work

A. Related Work on DNN and Path Search Problems

The use of a Convolutional DNN to approximate the heuristic path function is presented in [5]. The DNN is trained on different 2D map images, with different types of obstacles, each one with training maps and testing maps. After training, the DNN is presented with a map image containing the environment obstacles and a goal position. An image is generated as a result, where each pixel represents the distance to the goal position. With the construction of such heuristic map, it is queried during the path search. Similar to this work, the heuristic estimates are obtained from the trained DNN with a set of pre-computed paths.

A DNN architecture to predict the distance and time of taxi travels between two geographic coordinates according to the time of the day is presented in [6]. One of the DNN architecture layers is responsible for predicting the distance between two points and the other is responsible for predicting the time. The DNN responsible for predicting time receives as input the outputs of the last DNN layer. Such last layer is responsible for the distance predictions, along with the time of the day information. The data set used in the DNN training comes from GPS paths generated from New York City taxis. In this present work, the DNNs are also applied to such regression problem in order to predict topographic-aware routes between map coordinates.

The authors in [7] propose an algorithm similar to A^*. They used a DNN as a multiplier on the A^* heuristic portion (1). The DNN is trained on a set of randomly produced two-dimensional maps. After training, given a map, it generates a coefficient that represents the difficulty of moving through the map. Although we do not use such kind of coefficient in the path cost formulation, our approach is similar to this work since the heuristic portion is replaced by the value predicted by the DNN.

A DNN alternative to the traditional distance functions used as the heuristic for the path planning is described in [8]. This DNN extracts information from map images for the heuristic learning used in the planning of agent routes. The *Dijkstra* algorithm is used in the construction of an optimized heuristic value that is used to support the DNN training. The A^* is one of the algorithms used to evaluate the heuristics learned by the DNN. Our work has similar goals despite the fact that a Feed Forward architecture is used to learn the heuristic function, given by topographic-aware path computations on a graph-based terrain representation.

B. Related Work on Topographic Path Planning Problems

The work reported in [10] presents a formulation of a heuristic for capturing agent (mobile robots) movements with lower energy costs. A graph was used to represent a DEM from a region with canyons with 1km^2 area, where each vertex represents a terrain point and has 8 edges for its neighbors. The terrain points have three coordinates, two position coordinates and one with elevation information. The inclination angle between two terrain points was given by the tangent arc between their inclinations. On the formulation of the heuristic algorithm, this angle is used in the formulation of the energy cost model for the travel between the graph vertices. The used cost function is the (1) - type, where g is given by multiplying the Euclidean distance in the $\mathbb{R}^3$ with the composition of the gravity and friction forces acting on the agent. The agent has two thresholds analyzed during its path computations, the maximum angle it can climb and the maximum angle it can descend without losing contact with the ground. The heuristic h is similar to the g. However, it is not infinite for angles above the climbing threshold, allowing the agent to climb a hilly terrain beyond its capacity, using zig-zag shaped paths. Similar to this work, a DEM is also used in ours, with approximately the same terrain coverage area.

The authors in [9] present a path planning algorithm for 3D game scenes. The scene is abstracted using a graph, where vertices represent polygons and edges represent the polygon adjacency's used in the scene rendering. The edges also have the adjacent polygons' normal vectors and the maximum object dimensions that can pass through the adjacent polygons. The heuristic used by the path search algorithm is the

Euclidean distance between two terrain representation nodes. The terrain inclination represented by the normal vectors is used in the g function composition, where a linear combination of these vectors represents the difficulty of moving between the o and n nodes. In this current work, a graph is also used to abstract the terrain, where the graph vertices store the pixel height information they represent. The local movement difficulty between neighboring vertices, expressed in the g function, does not use the terrain characteristics represented by the normal vectors. It uses the relief height difference. Similar to our work, the inclination between the neighboring vertices is used when calculating the distance between them. That is computed by the Euclidean distance in the $\mathbb{R}^3$.

In [11], the authors reduce the challenge of computing paths for agents on rough or uneven terrain to a graph search problem. The navigation map is represented by a triangular 3D structure of a mesh. A graph is composed of vertices and the triangle sides. The other graph is formed by the triangles and the triangle adjacency's that make up the mesh. Different cost layers are associated with this representation structure, such as normal vectors to vertices and triangles, or a distance estimate based on the height difference between vertices. The different cost layers are aggregated at the time of path computation. The used path search algorithms were A^* and *Dijkstra*. Similar to this work, the graph edges represent the distance between adjacent vertices.

IV. DNNs as Heuristic Functions

In this work, the terrain height was represented in each node of the terrain representation structure. With that, the relief inclination between the terrain nodes is obtained. This is the case of [10], where the terrain height is used to calculate distances in the $\mathbb{R}^3$, which is critical to determining the agent movement costs in multi-agent problems. Then the cost is used to make up the cost of crossing two nodes of the terrain representation structure. In this structure, nodes represent each pixel of the elevation model. The Euclidean distance in the $\mathbb{R}^3$ is used to compute g and h during the A^* path search process. In the end, the topographic path costs make up the data set used in the DNN training, where the DNN is used to approximate the topographic heuristic function.

A. Terrain Maps

The real-world maps containing relief information used in this work were built from a DEM, where the map images were obtained from [16]. This DEM model is used in the radiometric image correction process obtained by the remote sensing with synthetic aperture radar (SAR). The DEM is a digital representation of the Earth's surface according to the mathematical model (datum) used to project the Earth into a plane. The DEM is an image represented in the GeoTIFF format, where each pixel color represents the elevation of the point on the Earth's surface. According to this model, elevation and inclination attributes of the used terrain are obtained.

The chosen DEM was used to correct the ALP-SRP277832830 image, aiming to fix possible image distortions due to the used imaging equipment. This image is shown in Fig. 1a and the corresponding DEM in Fig. 1b, where lighter colors represent higher relief altitudes. It has dimensions of 6288×5656 pixels, and a 12.5m spatial resolution. Such DEM is a re-sampling of the data from the *Shuttle Radar Topography Mission* (SRTM), which has a 30m spatial resolution. Such kind of real-world image presents an uneven relief, allowing some terrain areas to be blocked for the agent's navigation, while others present a non-flat relief and are navigable. To our simulation goals, the terrain area blocking considers the image pixel with an height above a certain threshold.

To compute paths using the DEM, a non-directed graph representation structure was used. In it, each DEM pixel is associated with a graph vertex. The pixel has an identifier associated with its DEM position. DEM neighboring pixels have adjacent vertices in the constructed graph. The weight assigned to the edge between adjacent vertices represents the Euclidean distance between the respective DEM pixels. This distance calculation considers the height difference between the represented graph vertices. The greater this difference is, the greater the terrain distance is, and also the inclination between the vertices.

The size of the used maps was defined with 100×100 pixels, which is consistent with typical maps used in many agent-based simulations. With this definition, sections of the DEM were selected, with different percentages of free and blocked areas. For this, 3 different 100×100 pixel map selections with approximately 30%, 50% and 70% free terrain areas for path computations were selected to support our tests. In addition, the selected maps where the node blocking threshold did not make the resulting graph disconnected were selected. To check whether this graph was disconnected, a breadth-first search algorithm was used in order to determine if the number of visited vertices was equal to the number of graph vertices. An instance of the selected maps is presented in Fig. 2.

In summary, the preparation of the maps followed these steps: (i) cut the DEM into 100×100 pixels, using the *gdal_translate* [17] tool; (ii) represent the selected DEM map characteristics into the graph structure; (iii) vary the relief height threshold and verify the resulting percentages of free and blocked areas; and (iv) check if the relief height threshold made the graph disconnected (i.e. to avoid to have map regions that are non-accessible for the path planning computations).

B. Training and Testing Datasets

With the selected terrain maps, three data sets were constructed, one for each map. To generate them, 50% of the free map pixels were randomly selected for path computations. This process divided the map into four quadrants, permitting to select 25% of the pixels in each quadrant. This sought to reduce the resulting dataset size, which stored shortest topographic distances computed in each map. The dataset size is given by combining the number of selected pixels taken two by two. Table I summarizes the characteristics of the maps used in the construction of the DNN training and testing data.

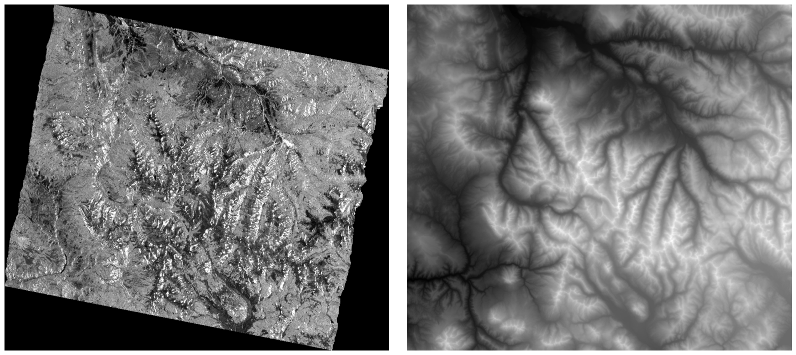
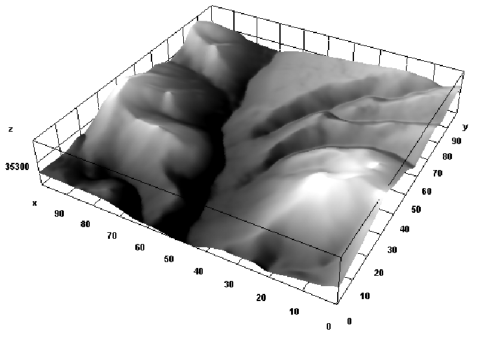
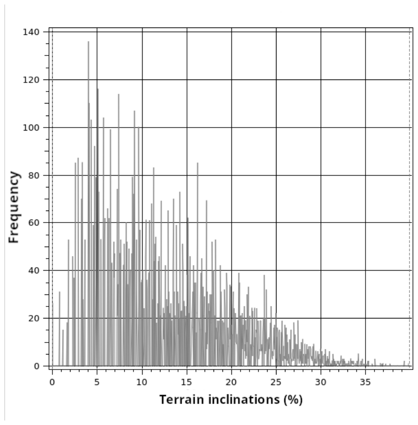

(a) (b)

Fig. 1: ALPSRP277832830 image - (a) Radiometric terrain corrected image, (b) DEM used to fix the image.

(a) (b)

Fig. 2: (a) 3D view of Map 1 and (b) histogram showing the distribution of Map 1 terrain inclinations.

TABLE I: DNN training information.

	Map 1	Map 2	Map 3	Maps 1, 2 and 3
Representation structure	square grid	square grid	square grid	-
Grid size	10^4	10^4	10^4	-
Blocked pixels	6,703 (67%)	5,009 (50%)	2,910 (30%)	-
Free pixels	3,297 (33%)	4,991 (50%)	7,090 (70%)	-
Height terrain threshold	2,504 m	2,788 m	3,506 m	-
Minimum terrain height	2,463 m	2,679 m	3,345 m	2,463 m
Maximum terrain height	2,601 m	2,861 m	3,614 m	3,614 m
Sample	1,648 (50%)	2,495 (50%)	3,545 (50%)	-
Dataset size	1,357,128	3,111,265	6,281,740	10,750,133
Dataset generation	7h	19h	45h	-
Train/test/validation sets	70/15/15	70/15/15	70/15/15	70/15/15

With the selection of map pixels, topographic paths were computed for all point combinations taken two by two (indicating the start and destination of the paths) in each used map. To compute the path for the datasets, the A^* algorithm was used with the Euclidean distance in the $\mathbb{R}^3$ as a heuristic h and the cost function g (Equation 1). That is because the value of g is equal to the edge weight between adjacent vertices. For each computed path, the origin and destination points in the map, and the topographic-aware path cost between them were stored in a text file using the following line format: origin, destination, path length. Another dataset was also built from the combination of the individual ones built for each map used on our tests. In this larger dataset, it was added an addition parameter in the line format to represent the identity of the map in which the path was computed. In the end, large numbers of path costs were used as input in the training of the DNNs.

C. DNN Architecture and Training

All DNN implementations used the Python language v3.73, the TensorFlow library v2.4.0, and the Keras libraries. As initial DNN architectures, the tested models had six inputs (since the origin and destination points in the datasets have three coordinates), an output that uses the linear function as the activation function (since the datasets were not normalized), and a varied number of hidden layers with a symmetric number of neurons in relation to the central hidden layer. To carry out tests with DNN architectures, the datasets were separated into three subsets, selected using random seeds: training, validation and testing. The training set has 70% of the dataset and the validation and testing sets have 15% each.

This work uses the Gradient Descent method to minimize the error function used in the DNN training. The error was estimated as each dataset instance was propagated by the DNN, after all instances were propagated by the DNN, or after a batch of instances was propagated by it. With a frequent error estimation method and small batch sizes, it was possible to avoid local minimum during the error function minimization process, but it slowed down the DNN training. We used batch sizes larger than one and smaller than the DNN training set cardinality, seeking a balance between training time and required computational resources. As the used batch sizes were powers of 2 when graphic processing units were used in the DNN training, the sizes of the tested DNN batches were: $2^k, k = 9, \ldots, 13$.

To stop the training, the limit of 2 thousand epochs or 100 epochs without decreasing the error in the validation dataset was empirically determined. That was expressed by the mean absolute percentage error (MAPE) [18] given by (2), where $y_t - \hat{y}_t$ 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 accuracy. The lower the value returned by the MAPE function is, the better the fit of the trained DNN model is. These limits were established to limit the maximum training time. After the training was stopped, the settings with the lowest MAPE (2) for each model, batch size and map had the DNN accuracy measured in the test dataset. To do so, we used the MAPE function (2).

$$\text{MAPE} = \frac{100}{n} \sum_{t=1}^{n} \left| \frac{y_t - \hat{y}_t}{y_t} \right| \tag{2}$$

In all cases, the model with the [6, 500, 100, 200, 300, 400, 500, 400, 300, 200, 100, 20, 1] number of neurons per layer, a batch size greater than or equal to 2,048, linear activation function in the output layer and a ReLU activation function in the hidden layers, obtained the best accuracy. To support the use of the much larger fourth dataset in the DNN training, the first layer of the chosen DNN model was changed from 6 to 7 neurons, permitting to identify which map the computed path data came from. The characteristics of the used DNNs in each terrain map are presented in Table II.

The trained DNN was used to replace the originally employed heuristic function h on the total path cost f (1). Along with the A^* algorithm, the trained DNN was tested as the heuristic function replacing the h function as shown in (4) and (5). There, d is the Euclidean distance between two points, dnn the DNN prediction of the distance between two points. The adopted A^* algorithm considered the terrain topography, which was captured by the graph representation, where the graph edge weight between adjacent vertices was defined as the Euclidean distance in the $\mathbb{R}^3$ among these vertices.

TABLE II: DNN training results.

	Accuracy	Training time	Epochs	Batch size
Map 1	98.22%	50 min	529	4096
Map 2	98.40%	80 min	413	8192
Map 3	99.04%	150 min	341	4096
Maps 1, 2 and 3	98.78%	623 min	796	2048

$$h_{r3}(n, m) = d(n, m) \tag{3}$$

$$h_{sub}(n, m) = dnn(n, m) \tag{4}$$

$$h_{subAll}(n, m) = dnn(n, m) \tag{5}$$

When the path planning computations need to be developed, the DNN predictions are calculated for all map positions, taken two by two, considering the free graph vertices that represent the DEM. The total time to calculate these predictions is divided by the number of predictions, resulting the time per prediction. Then the prediction time is added to the execution time of the A^* algorithm. In summary, with a graph representing the DEM of the terrain map, 50% of the graph nodes or vertices were selected. Then, paths were computed using the A^* algorithm with the Euclidean distance in the $\mathbb{R}^3$ as the heuristic function. These computed paths were used in the DNN training, validation and accuracy tests. On the other hand, paths were computed with the A^* algorithm using the Euclidean distance on the $\mathbb{R}^3$ in 7% of the graph nodes. In these 7%, paths were also computed with the results of the trained DNN acting as the heuristic function. For the paths computed on the testing nodes, execution time and number of expanded nodes are compared.

V. Experiments and Results

The experiments aimed to evaluate whether the DNN-based heuristic method was able to learn the terrain relief geometry and the blocked terrain regions for supporting the agents' movement in a simulated map containing topographical information. Two broad approaches were tested: i) the DNN trained with path data from each map (three individual DNN as we used three different maps in the tests) and ii) the DNN trained with path data from all maps at the same time were used as a direct approximation of the heuristic function h of the A^* algorithm. These two approaches were evaluated because the use of a DNN to search paths in different maps represent an important asset when the simulations require such functionality, which is frequently the case in the SIS-ASTROS project. However, it is also relevant to explore just one terrain map under concern, when one needs to execute a particular simulation in a selected terrain region as requested by users.

For each terrain map used in the performed tests, paths with 3 different heuristic formulations were computed - (3), (4) and (5). They were computed in the same execution environment where the training datasets were obtained (Intel(R) Core i7-6700k, 4(8) cores, 4.0 GHz; NVIDIA GeForce GTX 1050Ti, 4GB; 2x8GB DDR4-2800; Debian stable). The heuristic used as the basis for the path planning evaluations is the Euclidean distance on the $\mathbb{R}^3$.

To analyze the behavior of the A^* algorithm in relation to the tested heuristic options, paths were computed between all possible combinations in 7% of the total vertices of the graph that abstracts each map. The vertice values are presented in Table III. Dataset sizes are proportional to the free map area that the agents move in the terrain. To perform the path planning analysis, the following metrics were used: the execution time of the topographic path planning algorithm; the number of nodes evaluated during the search; and the cost/distance of the computed paths.

TABLE III: Data for the experiments.

	Map 1	Map 2	Map 3
Sample	231 (7%)	349 (7%)	496 (7%)
Computed paths	26,565	60,726	122,760

To statistically analyze the results, linear regression models were used (details about these statistical models can be found in [19]). They are a generalization of linear regression models that allow response variables with a distribution different from the normal one. The runtime and number of expanded nodes variables are continuous and have an asymmetric relationship with the computed path distance (explanatory) variable. In addition, they have positive values. The *gamma* distribution is commonly used in this case. Thus, the generalized linear regression model used in the analysis was:

$$log(\mu) = \beta_0 + \beta_1 X + \beta_2 D_1 + \beta_3 D_2$$

where μ is the mean of the response variable distribution, X is the explanatory variable and D_i, $i = 1, 2$ are the *dummy* variables used to analyze the tested 2 DNN-based heuristics in the same regression model. For each heuristic, they assume a configuration shown in Table IV.

TABLE IV: Configuration of dummy variables.

Dummy variables	Heuristic function
$D_1 = 0$, $D_2 = 0$	h_{r3}
$D_1 = 1$, $D_2 = 0$	h_{sub}
$D_1 = 0$, $D_2 = 1$	h_{subAll}

To develop the statistical analysis, a significance level of $\alpha = 1\%$ was defined. The null hypothesis $\mathbb{H}_0$ and alternative $\mathbb{H}_1$ were defined as $\beta_i = 0$ or $\beta_i \neq 0$, for $i = 0, 1, 2$. If $\alpha >$ p-value, then $\mathbb{H}_0$ is not rejected. $\beta_1 > 0$ means that the response variable is directly proportional to the explanatory variable. $\beta_{i+1} > 0$, $\beta_{i+1} = 0$ and $\beta_{i+1} < 0$ for $i = 1, 2$, respectively mean that the heuristic function associated with D_i is more efficient, equivalent or less efficient than the heuristic used as the basis for the comparisons. The results obtained in the path computations for the tested heuristics are presented in Tables V and VI. The percentage values of node expansion and runtime for all tested Maps and heuristic functions are presented in Table VII. Fig. 3 displays the graphs for each tested heuristic function.

TABLE V: Statistical results x visited nodes for Maps 1, 2 and 3.

	Heuristic	Estimate	Std. Error	t value	$P(>\|t\|)$
Map 1	h_{r3}	0.0027	0	421.7	<2e-16
	h_{sub}	-1.6770	0.0065	-266.7	<2e-16
	h_{subAll}	-2.1890	0.0065	-337.3	<2e-16
Map 2	h_{r3}	0.0022	0	652.5	<2e-16
	h_{sub}	-1.9830	0.004	-497.8	<2e-16
	h_{subAll}	-2.6730	0.004	-672.1	<2e-16
Map 3	h_{r3}	0.0021	0	877.8	<2e-16
	h_{sub}	-2.2810	0.0028	-808.0	<2e-16
	h_{subAll}	-2.8400	0.0028	-1010.5	<2e-16

TABLE VI: Statistical results x runtime for Maps 1, 2 and 3.

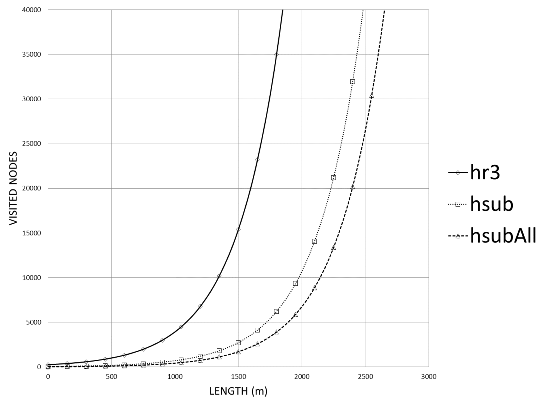
	Heuristic	Estimate	Std. Error	t value	$P(>\|t\|)$
Map 1	h_{r3}	0.0028	0	179.50	<2e-16
	h_{sub}	-0.8994	0.0017	-53.41	<2e-16
	h_{subAll}	-1.2980	0.0017	77.19	<2e-16
Map 2	h_{r3}	0.0022	0	266.8	<2e-16
	h_{sub}	-1.2110	0.0011	-110.0	<2e-16
	h_{subAll}	-1.7710	0.0011	-161.2	<2e-16
Map 3	h_{r3}	0.0021	0	397.5	<2e-16
	h_{sub}	-1.4030	0.0075	-185.9	<2e-16
	h_{subAll}	-1,9890	0.0075	-264.6	<2e-16

TABLE VII: Percentage of node expansion and runtime for Maps 1, 2 and 3.

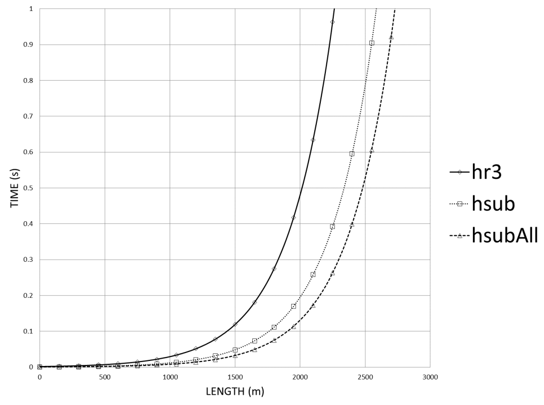
	Heuristic	Node expansion	Runtime
Map 1	h_{sub}	-82.27	-50.16
	h_{subAll}	-88.80	-63.36
Map 2	h_{sub}	-86.23	-64.69
	h_{subAll}	-93.10	-77.21
Map 3	h_{sub}	-89.78	-71.59
	h_{subAll}	-94.16	-82.48

Map 1 has 30% of its area free for agents' movement. The negative values for the regression model parameters when the number of expanded nodes is considered as an explanatory variable show that computing paths using the Euclidean distance expands a larger number of nodes in relation to other compared heuristics. This indicates that from the pre-computed path data in Map 1, which is used in the DNN training, the DNN learned the geometry and the blocked regions of the map, being able to generalize it in the heuristic computations. However, the order of magnitude of the estimates is different. For example, the h_{sub} estimate is -0.8994 while that of h_{subAll} is -1.2980. Thus, the A^* algorithm using h_{subAll} as a heuristic function expands the smallest number of nodes in relation to A^* with h_{r3} in -88.80%.

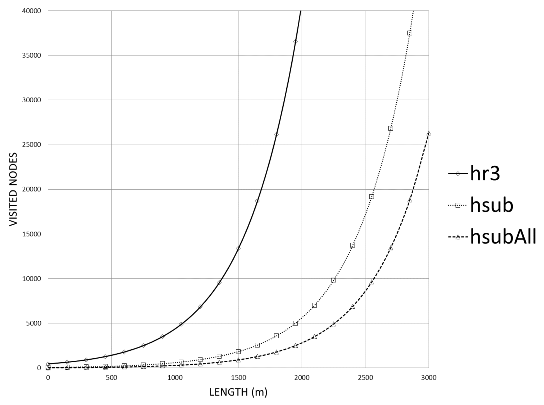
When the considered explanatory variable is the execution time, the regression model estimates for Map 1 show that using h_{subAll}, the A^* computes paths in smaller search times. In contrast to Map 1, 50% of Map 2 area is free for the agents' navigation. The results for the regression model parameters show that the increase in the free area did not change the behavior of the tested heuristics. The h_{sub} and h_{subAll} heuristics



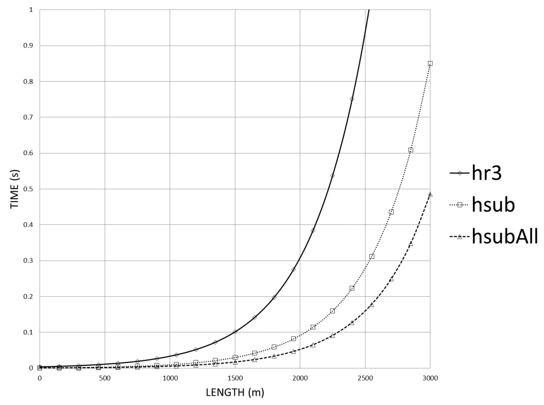
(a)



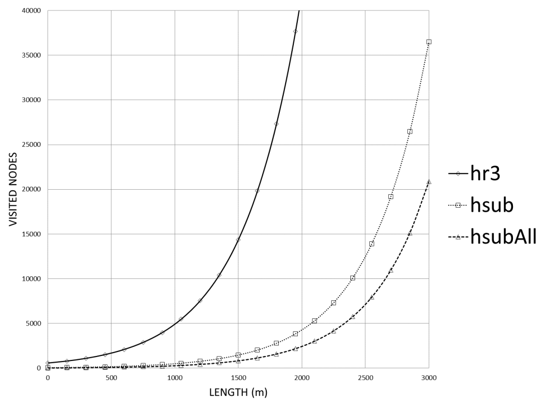
(b)



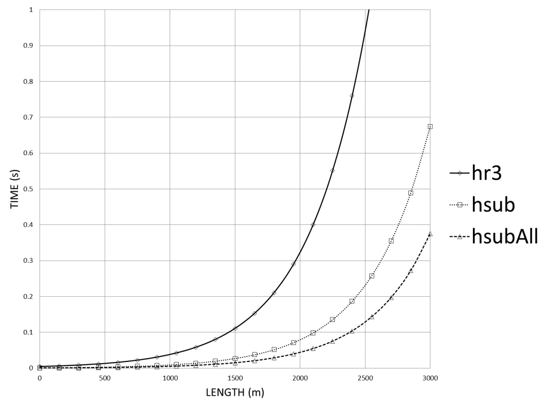
(c)



(d)



(e)



(f)

Fig. 3: Visited nodes X path length for (a) Map 1, (c) Map 2 and (e) Map 3; Runtime X path length for (b) Map 1, (d) Map 2 and (f) Map 3.

improved their performance compared to Map 1, expanding fewer nodes and using a shorter search time. Map 3 has 70% of its area free for agents' navigation. The number of paths computed in this case was $122,760$. The experimental results for Map 3 show the same behavior obtained in Maps 1 and 2. This suggests that the increase in the map free area does not interfere in the DNN learning.

The absolute mean percentage error (MAPE) function was used (Function 2) to evaluate how much the paths produced by the $A^\star$ using the heuristics h_{sub} and h_{subAll} have a different cost than the paths computed by the $A^\star$ with the heuristic h_{r3}. The results (Table VIII) showed that when replacing the heuristic h_{r3} with the DNN prediction, the $A^\star$ algorithm computed paths with an average difference of up to 0.5%.

TABLE VIII: Differences between computed paths.

	Map 1	Map 2	Map 3
h_{sub}	0.25%	0.50%	0.41%
h_{subAll}	0.12%	0.27%	0.18%

To sum up, the results regarding the use of the DNN trained with data coming from all tested real-world maps showed lower execution times and numbers of expanded nodes than the results obtained when a different DNN was trained and used in the heuristic computations of each map. It means that one can collect a likely set of terrain maps needed for the simulation executions. Then these maps can be used in the generation of topographic path data for the training of a single DNN with higher generalization capabilities. Whenever the simulation users require, however, particular terrain maps can also be used as input for the DNN training aiming to prepare the simulations to be executed in a particular terrain scenario. One possible drawback of this approach is related to the length of returned paths computed with these DNN-based heuristic functions. Our experiments show that this length was up to 0.5% longer than the shortest possible topographic-aware paths. That is a small price to pay when one needs to optimize the path planning computations in order to improve the smoothness and visual realism of the simulations. Although the integration of DNN and path planning techniques can be investigated in different ways, this work shows that the direct replacement of the traditional heuristic by the DNN prediction permits to reduce the computational cost of the path planning algorithm when it explicitly uses the topographic characteristics of the simulated terrains.

VI. Concluding Remarks

Deep Learning along with Path Planning are crucial AI techniques to the development of simulation systems. To the enhancement of the Agent-Based Modelling and Simulation paradigm, this work investigates how to apply DNNs as heuristic functions for the A^* algorithm in order to deal with real-world simulated terrains containing topographical information. That is a relevant approach to the need of having realistic and optimized topographic path planning for simulated agents in different kinds of simulation applications.

Experimental results obtained in this work indicate that the DNNs are able to generalize the topographic knowledge learned from a sample of map points and their shortest topographic path costs computed in the time the maps are pre-processed as part of simulation preparation procedures, producing paths close to the paths obtained by the A^* algorithm with the heuristic h_{r3}. Moreover, the increase in the free map area extends the DNN training time in a non-linear way. However, this increase improves the DNN performance so that the A^* algorithm using the DNN prediction as the heuristic value expands less nodes and has a shorter execution time compared to the base algorithm. In all tested maps there was a reduction in the execution time and the number of expanded nodes when the DNN was applied to replace the heuristic portion of the A^* algorithm. All in all, these results show that enhanced agent-based simulation systems can take advantage from the used of the investigated approach for the integration of Deep Learning and Path Planning.

Future work may investigate other ways to use DNNs for capturing other features relevant to the agents' navigation actions along with the represented terrain relief. Further work with much larger terrain maps, terrains with other topographic characteristics, and other hierarchical search and representation strategies may also reveal other applications for the proposed approach. Particularly, the investigation presented here can benefit a number of real-world applications involving autonomous agents and simulation systems.

References

[1] C. Macal, "Everything you need to know about agent-based modelling and simulation," *Journal of Simulation*, vol. 10, pp. 144–156, 2016.
[2] M. S. B. Othman and G. Tan, "Enhancing realism in simulation through deep learning," in *2019 Winter Simulation Conference (WSC)*. IEEE, 2019, pp. 2795–2806.
[3] I. Goodfellow, Y. Bengio, A. Courville, and Y. Bengio, *Deep learning*. MIT press Cambridge, 2016, vol. 1.
[4] Z. Abd Algfoor, M. S. Sunar, and H. Kolivand, "A comprehensive study on pathfinding techniques for robotics and video games," *International Journal of Computer Games Technology*, vol. 2015, 2015.
[5] Y. Ariki and T. Narihira, "Fully convolutional search heuristic learning for rapid path planners," *arXiv preprint arXiv:1908.03343*, 2019.
[6] I. Jindal, T. Qin, X. Chen, M. Nokleby, and J. Ye, "A unified neural network approach for estimating travel time and distance for a taxi trip," *arXiv preprint arXiv:1710.04350*, 2017.
[7] G. Li, G. Wang, Q. Wang, F. Fei, S. Lü, and D. Guo, "Ann: A heuristic search algorithm based on artificial neural networks," in *Proceedings of the 2016 International Conference on Intelligent Information Processing*, 2016, pp. 1–9.
[8] T. Takahashi, H. Sun, D. Tian, and Y. Wang, "Learning heuristic functions for mobile robot path planning using deep neural networks," in *Proceedings of the International Conference on Automated Planning and Scheduling*, vol. 29, 2019, pp. 764–772.
[9] S. Chen, G. Shi, and Y. Liu, "Fast path searching in real time 3d game," in *WRI Global Congress on Intelligent Systems*, vol. 3, 2009, pp. 189–194.
[10] N. Ganganath, C.-T. Cheng, and K. T. Chi, "A constraint-aware heuristic path planner for finding energy-efficient paths on uneven terrains," *IEEE Trans. on Industrial Informatics*, vol. 11, no. 3, pp. 601–611, 2015.
[11] S. Pütz, T. Wiemann, J. Sprickerhof, and J. Hertzberg, "3d navigation mesh generation for path planning in uneven terrain," *IFAC-PapersOnLine*, vol. 49, no. 15, pp. 212–217, 2016.
[12] S. Abar, G. K. Theodoropoulos, P. Lemarinier, and G. M. O'Hare, "Agent based modelling and simulation tools: A review of the state-of-art software," *Computer Science Review*, vol. 24, pp. 13–33, 2017.
[13] SIS-ASTROS, "Simulation project astros 2020 - project 3.07.0065/agreement 813782/2014." Brazil: Federal University of Santa Maria, Education Ministry, 2014.
[14] J. R. Brondani, L. A. de Lima Silva, E. Zacarias, and E. P. de Freitas, "Pathfinding in hierarchical representation of large realistic virtual terrains for simulation systems," *Expert Systems with Applications*, vol. 138, p. 112812, 2019.
[15] C. Chagas, E. Zacarias, L. A. de Lima Silva, and E. Pignaton de Freitas, "Hierarchical and smoothed topographic path planning for large-scale virtual simulation environments," *Expert Systems with Applications*, vol. 189, p. 116061, 2022.
[16] A. DAAC, "Palsar_radiometric_terrain_corrected_high_res; includes material ©jaxa/meti 2007," 2014.
[17] GDAL/OGR contributors, *GDAL/OGR Geospatial Data Abstraction software Library*, Open Source Geospatial Foundation, 2021. [Online]. Available: https://gdal.org
[18] A. De Myttenaere, B. Golden, B. Le Grand, and F. Rossi, "Mean absolute percentage error for regression models," *Neurocomputing*, vol. 192, pp. 38–48, 2016.
[19] P. McCullagh and J. A. Nelder, *Generalized Linear Models*. CRC Press, 1989, vol. 37.