UDC 681.3. **doi: 10.32620/reks.2026.1.12**

Andrii LIASHENKO, Larysa GLOBA

National Technical University of Ukraine «Igor Sikorsky Kyiv Polytechnic Institute», Kyiv, Ukraine

ACCELERATING A* ALGORITHM BASED SEARCH IN TIME-DEPENDENT GRAPHS WITH LEARNED HEURISTICS

Finding the shortest path in busy urban transport networks represented as time-dependent graphs is a complex task that demands algorithms with low computational complexity. Static heuristics based on geometric distance do not account for traffic dynamics, which limits the effectiveness of the A algorithm. The subject of the study is methods for increasing the efficiency of the A* search algorithm by creating heuristic functions based on machine learning. The goal of the work is to reduce the number of node explorations of the A* algorithm in a time-dependent graph by training the heuristic function. The research tasks are to conduct a comparative analysis of machine learning models that can be used as a heuristic function (LightGBM, GNN, TGNN); develop a hybrid method where TGNN predicts the future physical state of the system; and experimentally evaluate the efficiency gain. The main contribution is a hybrid approach where a Temporal Graph Neural Network (TGNN) predicts the future speed at neighboring nodes, which is then integrated into a classical geometric formula to compute an admissible and consistent heuristic. This resolves the contradiction between the high average accuracy of ML forecasts and the strict requirements for A* consistency. The models were evaluated based on three key criteria: admissibility, consistency, and search space reduction. The best model was selected as the one satisfying all criteria while achieving the greatest reduction in search space. The model was trained on real GPS data and tested on the road graph of the city of Irpin (2,400 nodes, 5,600 edges). The TGNN-based heuristic reduced the search space by 41.8% compared to Dijkstra and by 27.9% compared to the classical heuristic, with an average execution time of 0.44–0.65 ms per query and a 99.92–100% optimality rate across multiple routing scenarios. Conclusions. The scientific novelty lies in proving that reformulating the ML model's task from predicting total travel time to predicting a physical parameter (speed) effectively resolves the contradiction between ML forecast accuracy and A* consistency requirements, enabling the creation of adaptive heuristic functions for time-dependent pathfinding.*

Keywords: *A* algorithm; time-dependent graph; machine learning; shortest path search; temporal graph neural networks TGNN.*

1. Introduction

1.1. Motivation

Modern urban transport networks are constantly under strain due to dynamic changes in demand, traffic distribution and unpredictable congestion. According to the INRIX Global Traffic Scorecard (2024), the average duration of traffic jams in European cities has increased by 15% compared to the previous year, which creates additional pressure on logistics and navigation systems [1]. This requires the implementation of modern IoT systems that can manage traffic dynamically, taking into account changing traffic conditions. However, such systems require algorithms that have low computational complexity. In such conditions, the task of finding the shortest path becomes significantly more difficult, especially when the weight of the edges of the graph depends on time (time-dependent graph). This requires not only an accurate forecast of the state of the graph, but also an efficient search algorithm that can adapt to changes in real time.

In this context, route optimization in dynamic traffic environments requires not only the prediction of travel times but also adaptive methods that can efficiently update routing decisions when new information becomes available. Therefore, the development of intelligent pathfinding algorithms that combine classical search principles with machine learning approaches becomes a promising research direction for smart transportation systems.

One of the most promising approaches to routing is the use of the A* algorithm, which significantly reduces the volume of computations by employing a heuristic function h(u). The A* algorithm is one of the most popular and widely used search algorithms for finding the shortest paths in graphs, particularly in routing, robotics, and computer games. Its popularity stems from the

ISSN 1814-4225 (print)
ISSN 2663-2012 (online)

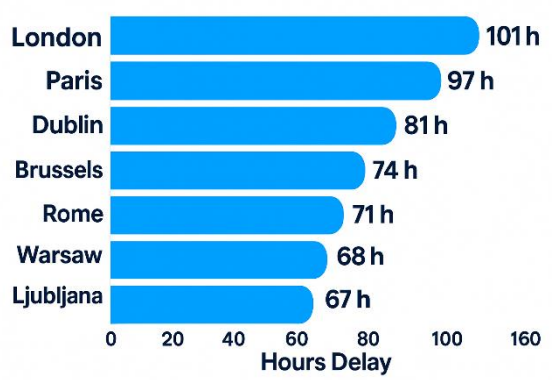

Fig. 1. Average time lost in traffic jams in European cities [1]

combination of Dijkstra's accuracy with the speed of greedy search. A* performs a search from the starting node to the target node, prioritizing nodes that minimize the function:

$$f(u) = g(u) + h(u), \quad (1)$$

where $g(u)$ is the actual cost (distance or time) from the starting node to the current node u;

$h(u)$ is the heuristic estimate (approximate cost) from node u to the target node [2].

However, in the case of time-dependent graphs, where the weight of edges (travel time) changes depending on the departure time, the effectiveness of A* faces a fundamental contradiction (e.g. straight-line distance or fixed speed) [3]. The algorithm's efficiency and optimality are guaranteed only if the heuristic function satisfies certain mathematical properties, namely admissibility and consistency. The classical heuristic, based on geometric distance, is overly simplified as it completely ignores the actual traffic state, even though it meets these requirements. This leads to a decrease in its effectiveness under dynamic conditions.

To resolve this contradiction, this work proposes the concept of finding the heuristic function $h(u)$ for transport networks with dynamic traffic conditions using machine learning methods on historical data. The research is focused on the influence of the heuristic function on the speed and computational efficiency of A*, particularly on the size of the search space explored.

1.2. State of the art

Route optimization in time-dependent graphs is a critical problem for both urban infrastructures and specialized telecommunication and transport networks. A* search remains one of the most effective algorithms in this field due to its ability to reduce computations using a heuristic function. However, classical heuristics are based on static estimates (Euclidean distance, straight-line distance), which do not account for changes in edge weights over time caused, for example, by traffic congestion or graph dynamics.

Existing research offers various approaches to account for temporal changes. The work in [4] addresses the problem of LEO satellite network topology optimization, where models considering temporal changes are applied, which is close to our case of a dynamic graph, but the focus is on topological planning rather than routing. In [5], the authors apply graph optimization for manufacturing tasks where time plays a key role, but the method does not focus on reducing computations for A* search. A classical approach to accelerating search is goal-directed search with precomputed heuristics, as in [6], but it only works in static graphs and is not suitable for time-varying scenarios.

Recently, deep learning models for ETA prediction have gained significant attention. For instance, [7] introduced the DeepETA model, a spatio-temporal recurrent neural network that learns from logistics data. Although this approach was not directly integrated into search algorithms like A*, it is a promising candidate for use as a dynamic heuristic. It is this idea the construction of ML-heuristics for A* search in time-dependent graphs that is implemented in this study.

Recent research continues to advance the integration of deep spatio-temporal graph models for intelligent routing. For instance, [8] proposed a multiscale spatio-temporal GNN (STGMS) that captures both regional and global dependencies in traffic forecasting, achieving state-of-the-art performance in dynamic congestion prediction. Similarly, [9] introduced the Adaptive Spatio-Temporal Dynamic Graph Convolutional Network (AST-DGCN), which dynamically updates graph structure to reflect real-time traffic variations.

These findings emphasize the necessity of combining graph topology learning with temporal modeling – the direction that this study continues.

For an adequate modeling of such systems, the concept of a time-dependent graph [8] is employed.

A time-dependent graph $G(V,E,C)$ is a tuple, where

V is a finite set of vertices (nodes), representing, for example, intersections.

E is a set of edges, representing road segments.

C is a set of cost functions for each edge, which defines the cost (travel time) of traversing from node u to node v when departing at time t.

A key element of the model is the cost function c_{uv} It can be represented as tabular data, a simple mathematical function, real-time traffic data (e.g., GPS

probes, IoT sensors, or live traffic feeds), or any other time-dependent model describing the travel time along edge (u, v) when departing at time t. This function reflects the actual travel cost and is not a heuristic for shortest path algorithms to operate correctly, the cost functions must satisfy the FIFO property. This property ensures that arrival at node v cannot occur earlier if departing later from node u [11].

The main task in such a graph is to find the fastest path. The arrival time at node v, denoted as $\mathbf{T}_{\text{arrive}}(\mathbf{v})$, can be defined by a recurrence relation that is an adaptation of the Bellman equation for the time-dependent case [12]:

$$\mathbf{T}_{\text{arrive}}(\mathbf{v}) = \min_{u \in \text{Pred}(v)} \{\mathbf{T}_{\text{arrive}}(u) + c_{uv}(\mathbf{T}_{\text{arrive}}(u))\}, \quad (2)$$

where $\mathbf{T}_{\text{arrive}}(\mathbf{v})$ is the earliest possible arrival time at node v;

$\text{Pred}(v)$ is the set of all nodes u from which there exists a direct edge to v;

$\mathbf{T}_{\text{arrive}}(u)$ is the earliest arrival time at the predecessor node u;

$c_{uv}(\mathbf{T}_{\text{arrive}}(u)$ is the travel cost along edge (u, v) when departing from u at the time of arrival at u.

This recurrence relation forms the theoretical foundation of time-dependent shortest path algorithms. In algorithms such as time-dependent Dijkstra or A*, a **heuristic function** $h(v)$ can be additionally used to guide the search toward the target. Modern approaches employ machine learning models, such as **Temporal Graph Neural Networks (TGNNs)** [13], specifically for constructing this heuristic. TGNNs are able to predict the expected remaining travel time (ETA) from an intermediate node v to the destination, thereby improving the efficiency of time-dependent A*.

This paper proposes the construction of an adaptive heuristic function $h(u)$ for the A* algorithm, which considers both the graph structure and the temporal characteristics of the edges. An initial hypothesis was the direct prediction of the final arrival time (ETA) using powerful ML models (LightGBM, GNN). However, experiments showed that this direct approach is ineffective because the resulting heuristics were inconsistent, leading to a deterioration, rather than an improvement, in A* search efficiency.

Therefore, instead of a naive direct estimation method, this paper proposes a hybrid approach. Its essence lies in changing the task for the model: a Temporal Graph Neural Network (TGNN) is used to predict not an abstract cost, but a concrete physical parameter the future speed $v_{\text{TGNN}}(u, t)$ at a neighboring node. This prediction is then integrated into a mathematically stable classical formula that uses the great-circle distance $D_{\text{phys}}(u, d)$ to determine the arrival time. A combined loss function, including MSE and a consistency loss, is used to train this model to ensure the heuristic function has the property of admissibility.

A key requirement is admissibility, according to which the heuristic should never overestimate the true cost of the path to the goal $h(n) \le h^*(n) \forall n$. A more stringent requirement is consistency, or monotonicity, which requires the triangle inequality $h(u) \le c(u, v) + h(v)$ to hold for any two adjacent nodes and guarantees that the cost estimate is logically consistent along the path.

1.3. Objectives and tasks

The objective of this work is to increase the computational efficiency (reduce the number of expanded nodes) of the time-dependent A* algorithm by developing and justifying a hybrid method for creating a heuristic function based on machine learning.

To achieve the goal, within the framework of this publication it is necessary to solve the following **tasks**:

1. To conduct a comparative analysis of the efficiency of the classical geometric heuristic and heuristics trained to directly predict ETA using LightGBM and static GNN models.
2. To develop a hybrid approach for creating a heuristic where a TGNN model predicts future travel speed, and the final value is calculated by integrating the prediction into a classical formula.
3. To experimentally test the proposed hybrid approach on real data and quantitatively assess its advantage over other methods based on search speed.
4. To analyze the results and formulate conclusions regarding the optimal strategy for applying ML to accelerate A* search in time-dependent graphs.

The article is structured as follows:

Section 2 "Statement of the research problem" formalizes the task, defining the input and output data in a way that is directly applicable to real-world scenarios. Section 3 «Analysis ML Models for Heuristic Selection» provides an overview of the selected models, highlights their strengths and limitations in the context of time-dependent shortest path problems, and justifies their applicability for building heuristic functions. Section 4 «Hybrid approach». Section 5 «Experiment» reports the experimental results and compares them with the

ISSN 1814-4225 (print)
ISSN 2663-2012 (online)

outcomes obtained using heuristics trained by other machine learning models. Section 6 concludes the article by summarizing the findings and outlining directions for future research.

2. Statement of the research problem

2.1. Input Data and Graph Construction

To address the problem of increasing the efficiency of finding the fastest path in a dynamic urban transport network, two main types of input data were used:

Static road graph. To obtain a topologically correct graph, data from the OpenStreetMap (OSM) project - a global open map that has become the de facto standard for academic transport research – were used [14]. The road network structure of the city of Irpin, extracted from OpenStreetMap with the help of the OSMnx library [15], was employed. This graph $G(V,E,C)$ consists of:

- **Nodes (V):** Intersections, which have static properties such as geographic coordinates.
- **Edges (E):** Road segments connecting nodes, which have static properties such as length.

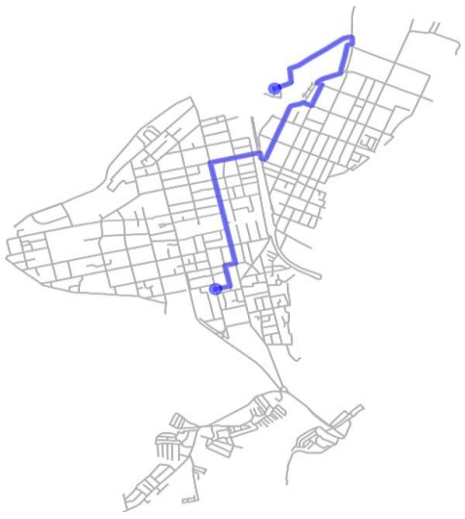

Fig. 2. Finding the shortest path using the OSMnx library

Dynamic traffic data. A real-world traffic dataset collected from GPS-equipped vehicles operating in the city of Irpin was used. The data were anonymized, aggregated at 5-minute intervals, and cover typical weekday traffic patterns including morning and evening rush hours. The dataset is available upon reasonable request from the corresponding author. These data are represented as a time series describing how the traffic situation on the roads changes throughout the day. Each record includes:

- v_i, v_j (Node identifiers): The start and end nodes of a road segment, corresponding to intersections that determine the vehicle's route.
- timestamp: The time at which the data were collected from sensors (data were received from sensors every 5 minutes).
- speed: The average speed along the edge in km/h.
- length: The length of the edge in meters.
- eta (Estimated travel time): The calculated travel time in minutes, spent to cover the distance between v_i and v_j, which corresponds to the edge weight.

For training the GNN and LightGBM models, a dataset was created using Dijkstra's algorithm. The dataset was generated by calculating the optimal travel time from thousands of random nodes to a single fixed destination node at different times of the day. Each row represents one training sample and consists of the following fields:

- $start\_node_x, start\_node_y$: Geographic coordinates of the starting node.
- $finished\_node_x, finished\_node_y$: Geographic coordinates of the destination node (the same for all records).
- $start\_time$: Trip start time.
- $distance$: Great-circle distance between the start and the destination.
- $ground\_truth\_eta$: The real optimal travel time, which served as the target variable for supervised learning.

For the LightGBM model, preprocessing of the data was performed, resulting in a separate feature vector. This vector provided a complete description of the specific shortest-path search task and included:

- **Geometric features**: Coordinates of the start and target nodes, as well as the great-circle distance between them.
- **Temporal features**: Trip start time.

The **ground truth eta** was used as the target variable for supervised learning.

2.2. Output definition

The expected output of the proposed approach is an estimate of the travel time from a neighboring node to the target node. This value is not a full path computation but rather a heuristic prediction that helps reduce the number of nodes explored by the A* algorithm. By providing an admissible and informative heuristic, the algorithm can prioritize more promising search directions and thus significantly improve computational efficiency in large-scale time-dependent graphs.

To achieve this objective, we propose to employ machine learning models capable of learning both spatial and temporal traffic dependencies. Gradient boosting methods such as **LightGBM** [16] serve as strong baselines for processing tabular features, while **Graph Neural Networks (GNNs)** [17] are effective for modeling relational structures of the road network by aggregating information from neighboring nodes. Building on this foundation, **Temporal Graph Neural Networks (TGNNs)** [18] extend GNNs with the ability to capture temporal dynamics, making them particularly well-suited for representing time-varying patterns of road traffic. The following section analyzes and compares these models with respect to their ability to generate heuristic functions, aiming to identify the most effective approach for guiding the A* search process in dynamic urban transport networks.

3. Materials and research methods

3.1. LightGBM

The first step was to evaluate the effectiveness of a powerful statistical model that does not take into account the graph structure directly. **LightGBM** was chosen – an implementation of gradient boosting on decision trees, known for its speed and high accuracy on tabular data [16].

Gradient boosting is an ensemble method that sequentially builds weak models (in this case, decision trees), where each subsequent model $F_{m+1}(x)$ is trained to correct the errors (residual gradients) of the previous ensemble $F_m(x)$. The process can be described as:

$$F_{m+1}(x) = F_m(x) + h_m(x), \quad (3)$$

where $h_m(x)$ is a new decision tree that minimizes the loss function (MSE). This approach made it possible to assess how well a **non-graph** model can approximate the heuristic function using only numerical node features.

Unlike LightGBM, graph neural networks take as input the entire graph and the feature matrix for all its nodes. The data preparation process consisted of two steps:

1. **Node feature construction.** For each node (intersection) in the graph, an initial feature vector was calculated. To provide the model with basic information about intersections, the average speed on all adjacent edges (roads) $speed_{mean}$ connected to this node was used as a feature. This allowed converting edge-level information into node-level features.

2. **Target definition.** The data were used to form the target vector and the training mask. For each node included in the training set, the ground truth value of travel time $ground\_truth\_eta$ was recorded, and the corresponding element in the mask was marked as trainable. Thus, the model learned to predict the correct heuristic only for nodes in the training set, while generalizing this knowledge to the entire graph.

3.2. Graph neural network

To directly account for the topology of the road network, graph neural networks (GNNs) were investigated. GNNs process data in their natural structure, allowing nodes to aggregate information from their neighbors [17]. The general principle of a GNN layer is described by the message passing mechanism, which consists of two stages (Fig. 3):

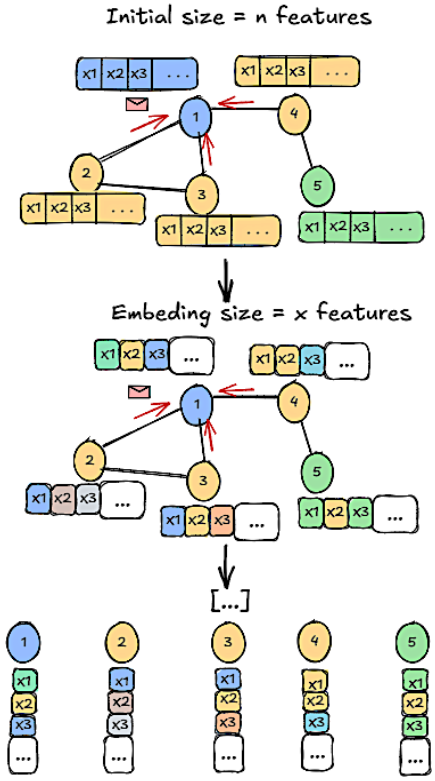

Fig. 3. Message Passing Mechanism in GNN

AGG (Information gathering): At this stage, each node «listens» to its neighbors. It collects their feature vectors (messages) and, using an aggregation function (e.g., mean, sum, or maximum), combines them into one aggregated message. This enables the node to obtain summarized information about the state of its local neighborhood.

UPD (State update): After receiving the aggregated message, the node updates its own state. It combines its current feature vector with the aggregated message, usually through a small neural network. As a result, the

ISSN 1814-4225 (print)
ISSN 2663-2012 (online)

node obtains a new, more informative embedding vector that now contains knowledge not only about itself but also about its neighbors.

Mathematically, this two-step process can be expressed as:

$$h_i^{(l+1)} = UPD^{(l)}(h_i^{(l)}, AGG^{l}(\{h_j^{(l)} : j \in N(i)\})), \quad (4)$$

where $h_i^{(l)}$ is the feature vector (embedding) of node i at layer l, and $N(i)$ is the set of its neighbors.

Two of the most influential inductive architectures for graph representation learning are GraphSAGE and Graph Attention Network (GAT).

GraphSAGE (Sample and aggreGatE): An inductive architecture that learns to create a universal aggregation function for neighbor features. A Mean Aggregator was used, with the update rule:

$$h_i^{(l+1)} = \sigma(W \cdot CONC(h_i^{(l)}, \frac{1}{|N(i)|} \sum_{j \in N(i)} h_j^{(l)})), \quad (5)$$

where $h_i^{(l+1)}$ is the new embedding vector for node i at the next layer $l + 1$.

σ is nonlinear activation function (e.g., ReLU).

W is weight matrix to be learned, essentially the neural layer that processes the combined information.

CONC is concatenation operation of two vectors.

$h_i^{(l)}$ is the current feature vector of node i.

$\frac{1}{|N(i)|} \sum_{j \in N(i)} h_j^{(l)}$ is aggregated messages from neighbors obtained by averaging their feature vectors. This makes GraphSAGE robust to changes in the graph [20].

Graph Attention Network (GAT): A more advanced architecture that uses an attention mechanism to dynamically weight the importance of neighbors during aggregation [21]. Attention coefficients α_{ij} are computed for each neighbor j of node i:

$$a_{ij} = \sigma_j(LeakyReLU(a^T[W \cdot h_i \,||\, W \cdot h_j])), \quad (6)$$

where a_{ij} is attention coefficient (importance) assigned by node i to node j;

W is learnable weight matrix for feature transformation;

h_i, h_j is feature vectors of nodes i and j;

$||$ is concatenation operator;

a^T is learnable parameter vector of the attention mechanism;

Leaky ReLU is nonlinear activation function.

The denominator σ is a softmax function that normalizes attention coefficients across all neighbors so that their sum equals 1.

The updated node representation is then computed as the weighted sum of its neighbors' transformed features:

$$h_i' = \sigma(\sum_{j \in (N(i))} a_{ij} W h_j), \quad (7)$$

where h' is the new feature vector for node i;

$\sum_{j \in (N(i))} a_{ij} W h_j$ is an aggregated message is a weighted combination of its neighbors' embeddings.

Both models were trained on an extended feature set that included geometric as well as topological characteristics of the graph.

3.3. Temporal graph neural network

TGNN analyzes sequences of graph «snapshots» over time, which allows it to account for traffic dynamics, such as congestion propagation [22].

The architecture is justified as follows. For the final and most successful experiment, an architecture was implemented that combines a Graph Convolutional Network (GCN) for spatial aggregation [19] and a Gated Recurrent Unit (GRU) for temporal sequence analysis [23].

Data processing in this architecture occurs as follows:

Input: The model receives as input a sequence of graph states from several previous time steps (e.g., $X^{(t-2)}, X^{(t-1)}, X^{(t)}$). Each such state, or "snapshot" is a feature vector for all graph nodes at a specific time.

Spatial processing (GCN): Each graph snapshot sequentially passes through a Graph Convolutional Network (GCN) layer. The task of the GCN is to capture the spatial structure of the graph at that moment. It aggregates information from neighboring nodes, creating intermediate spatial features $X'^{(t)}$ that reflect the local neighborhood of each node. A two-layer configuration was chosen as a justified compromise: the first layer aggregates information from immediate neighbors (1-hop), while the second layer extends the receptive field to neighbors of neighbors (2-hop), which is critical for capturing the local regional context [20]. However, using too many layers can cause oversmoothing [24].

Temporal processing (GRU): The sequence of spatial features $\{X'^{(t-2)}, X'^{(t-1)}, X'^{(t)}\}$ is passed to a Gated Recurrent Unit (GRU) layer. The GRU has

internal "memory" - the hidden state $H^{(t-1)}$, which stores information about previous system states. At each step, the GRU updates this hidden state by incorporating new spatial features. This enables the model to capture temporal dependencies and patterns (e.g., the growth of traffic congestion) [25].

Prediction: The final hidden state $H^{(t)}$ contains the combined spatio-temporal information. It is fed into a linear layer, which produces the final prediction (e.g., the average speed of each node at the next time step).

This architecture was chosen because it represents an effective standard for traffic state forecasting tasks. Its main advantage is the ability to simultaneously capture: how the state of a node depends on its neighbors in the current snapshot (via GCN), and how the state depends on its own historical evolution (via GRU).

Table 1

Comparative analysis of machine learning model approaches for finding heuristics

Method	Working Principle	Advantages	Limitations
LightGBM	Statistical ensemble of decision trees	1. High speed on tabular data 2. High accuracy	1. Does not exploit graph topology 2. May produce inconsistent predictions
GNN	Spatial feature aggregation	Captures graph topology	Does not account for temporal dynamics
TGNN	Spatiotemporal forecasting (GCN + GRU)	1.Captures topology and temporal dynamics	Highest computational complexity

3.4. Hybrid approach

Unlike baseline models, where the objective was to directly predict the travel time between the origin and the destination, such an approach proved insufficient for accurate heuristic estimation. Direct ETA prediction often fails to capture the evolving dynamics of the road graph under varying traffic conditions. Therefore, a hybrid TGNN-based strategy was adopted. Instead of producing ETA in a single step, the TGNN forecasts short-term traffic states (e.g., average speed on edges) by jointly modeling spatial dependencies through GCN and temporal dependencies through GRU. These predicted states are subsequently combined with the physical distance along the path to compute the estimated arrival time. This design ensures that the heuristic function remains both adaptive to traffic variations and consistent with the underlying graph structure, which is critical for effective guidance of the A* algorithm.

Previous experiments also demonstrated that training models solely for accuracy (e.g., minimizing MSE) leads to the creation of "noisy" and inconsistent heuristics, which degrade the efficiency of the A* search. This limitation arises because standard machine learning models do not explicitly account for the specific requirements of the A* algorithm, such as the admissibility and consistency properties of the heuristic. To address this issue, a **combined loss function** was introduced, forcing the model during training to respect these fundamental properties required by the A* algorithm.

The general loss function consists of two components:

$$L_{total} = L_{acc} + \lambda L_{cons}, \quad (8)$$

where L_{acc} **(Accuracy loss)** – This is the standard mean squared error (MSE), ensuring that the model prediction $h^*(n)$ is as close as possible to the true optimal travel time.

L_{cons} **(Consistency loss)** – This component penalizes the model for violating the triangle inequality (the property of consistency). For a randomly selected set of edges (u, v) from the graph, the loss is computed as the average magnitude of violations:

$$L_{cons} = \frac{1}{|E_{batch}|} \sum_{(u,v)} \max(0, h(u) - (c_{uv}(t) + h(v))) \quad (9)$$

where E_{batch} is denotes the subset of edges (u,v) sampled from the graph during a training batch, used to compute the consistency constraint.

$h(u)$ is the predicted travel time node u to the finished node.

$c_{uv}(t)$ is the transition cost from node u to v. This penalty forces the model to produce "smooth" and logically consistent predictions across neighboring nodes.

$h(v)$ is the predicted travel time from the neighboring node *v* to the same finished node

λ is a hyperparameter that regulates the trade-off between accuracy and consistency. Selecting the value of λ is a key compromise: small values prioritize accuracy but risk producing "noisy" heuristics, whereas large values enforce smoother and more consistent predictions but at the expense of accuracy. An empirically optimal

ISSN 1814-4225 (print)
ISSN 2663-2012 (online)

balance was found at $\lambda = 0.5$, which ensured the best trade-off between the two objectives. This approach enables the model to learn heuristics that are not only accurate but also mathematically "admissible" for effective A* guidance.

For training the Temporal Graph Neural Network (TGNN), a static temporal graph was used, meaning that the structure of the graph remained unchanged, while only the dynamics of traffic varied. The input data for the model consisted of a sequence of "snapshots" of the graph state, each corresponding to a 5-minute interval. Since the input data contained speeds aggregated at edges over discrete time intervals, it was necessary to determine the travel time eta at any arbitrary moment. Because traffic data are discrete (with a 5-minute step), the function applies linear interpolation. By taking the nearest known timestamps from neighboring nodes and calculating the weighted average of their ETA values, it was possible to obtain a more realistic continuous cost function for the A* algorithm. The linear interpolation formula is given as:

$$c(t) = c(t_1) + (c(t_2) - c(t_1)) \cdot \frac{(t - t_1)}{(t_2 - t_1)}, \qquad (10)$$

where $c(t)$ is interpolated travel cost (e.g., travel time or ETA) at an arbitrary timestamp t;

$c(t_1)$ is travel cost at the nearest known earlier timestamp t_1;

$c(t_2)$ is travel cost at the nearest known later timestamp t_2;

t is arbitrary query time for which the cost function value is required;

$\frac{(t - t_1)}{(t_2 - t_1)}$ is linear interpolation coefficient that defines how far the query time t lies between t_1 and t_2.

Each node feature (intersection) at every snapshot was represented by the average speed of movement $\text{speed}_{\text{mean}}$ over all edges connected to this node. The model was trained to perform forecasting tasks: based on the sequence of graph states over the previous period (60 minutes), it predicted the next state (speed vector) at the subsequent time step.

The key property of the TGNN model is that it captures not only spatial dependencies, i.e., how nodes are connected to each other, but also temporal dependencies. Thus, during A* execution, the neighbors of a node are explored with respect to both spatial and temporal context. The prediction of the arrival time to the destination works in such a way that the TGNN model can estimate the speed at future time intervals.

As TGNN passes information between neighbors and neighbors-of-neighbors during training (two internal layers were used), it becomes possible to predict the average speed at a node v at time t. By combining this prediction with the physical shortest distance between two points on the Earth's surface, the estimated arrival time to the destination can be defined using the formula:

$$h(u,t) = \frac{D_{\text{phys}}(u,d)}{v_{\text{TGNN}}(u,t)}, \qquad (11)$$

where $h(u,t)$ is estimated heuristic travel time (ETA estimate) from the current node u to the destination node d at time t;

$D_{\text{phys}}(u,d)$ is physical (geodesic) shortest distance between nodes u and d on the Earth's surface, typically computed using the Haversine formula or great-circle distance;

$v_{\text{TGNN}}(u,t)$ is predicted average speed at node u at time t, obtained from the Temporal Graph Neural Network model trained on historical and real-time traffic data.

A similar conceptual shift has recently been observed in studies on dynamic routing, where researchers argue that predicting interpretable physical quantities such as speed or flow provides more stable and transferable models than direct ETA regression.

For instance, [26] proposed a graph-based neural approximation for shortest-distance estimation under uncertainty, while [27] demonstrated that continual-learning-enhanced spatio-temporal GNNs can maintain prediction stability in non-stationary traffic environments.

These findings reinforce the rationale behind the hybrid TGNN model proposed in this study, which focuses on forecasting physical parameters rather than abstract travel-time values.

4. Experimental outcomes

The experiment was conducted on a time-dependent road graph of the city of Irpin, consisting of 2,400 nodes (intersections) and 5,600 edges (road segments), extracted from OpenStreetMap using OSMnx [15]. All experiments were executed on a Dell Latitude 5420 (Intel Core i7-1185G7, 32 GB RAM) using Python 3.13 with PyTorch, NetworkX, and NumPy. To ensure statistical robustness, the evaluation was repeated across 10 origin–destination pairs under three traffic conditions (morning rush at 8:00, midday at 13:00, evening at 18:00). The average execution time per query was 0.44–0.65 ms for the TGNN heuristic vs. 0.83–1.01 ms for Dijkstra, with 100% success rate and 99.92–100% optimality.

Six main approaches were evaluated:

1. **Simple heuristic:** A classical geometric heuristic computed as $h(u,t) = \frac{D_{phys}(u,d)}{v(u,t)}$

This approach guarantees admissibility and consistency but is insensitive to the actual traffic situation, since it assumes a constant maximum speed across the entire network and ignores local congestion or slowdowns.

2. **LightGBM heuristic:** A heuristic based on the LightGBM model.

3. **GNN SAGE heuristic:** A heuristic based on the GraphSAGE model.

4. **GATv2 heuristic:** A heuristic based on the more advanced Graph Attention v2 architecture.

5. **TGNN (hybrid) heuristic:** A heuristic derived from speed forecasts produced by the trained TGNN model.

6. **Dijkstra baseline:** Search without a heuristic (h=0), which serves as an upper bound on the number of explored nodes.

The models were evaluated based on three criteria: (1) admissibility – the heuristic must never overestimate the true cost; (2) consistency – the triangle inequality must hold for adjacent nodes; and (3) search efficiency – the number of explored nodes should be minimized. The best model was selected as the one satisfying all criteria while achieving the greatest search space reduction. Each model was evaluated on identical origin–destination pairs under the same traffic conditions.

The experimental results are presented in Table 2 and Figure 4.

Table 2

Comparative performance of heuristic models in dynamic shortest path search

Method	Path quality (min)	Search speed (explored nodes)	Reduction relative to Dijkstra
Dijkstra (h=0)	16.50	452	0%
GNN SAGE	16.50	528	-16.8
GATv2	16.50	446	1.3%
LightGBM	16.50	436	3.5%
Simple heuristic	16.50	389	13.9%
TGNN (1 GCN layer)	16.50	275	39.2%
TGNN (2 GCN l-rs)	16.50	263	41.8%

As can be seen from Table 2, all approaches successfully identify the optimal path with a travel cost of 16.50 minutes. However, there are significant differences in their search efficiency.

The weakest results were shown by static GNN models (SAGE and GATv2). Despite their ability to capture graph topology, they produced "noisy" and inconsistent heuristics, which caused the A* algorithm to explore even more nodes than the "uninformed" Dijkstra baseline. This indicates that static graph representations are insufficient for modeling dynamic road conditions where traffic flow varies over time. In particular, their heuristics often underestimated the travel time in congested regions, leading to frequent re-expansions of nodes and degraded performance.

The best performance was demonstrated by the TGNN-based heuristic trained with two GCN layers, which required exploring only **263 nodes**. This is **41.8% fewer** than Dijkstra and **27.9% fewer** than the simple heuristic. Moreover, the TGNN-based model provided smoother and more temporally consistent predictions of travel time, which improved both convergence speed and route stability. The results confirm that incorporating temporal dependencies allows the heuristic to generalize better across varying traffic states, effectively balancing accuracy and computational efficiency during pathfinding.

Fig. 4. The shortest path is found by the A* algorithm and TGNN as a heuristic

The experimental results confirm that naively replacing the classical heuristic with a predictive ML model does not guarantee improved search efficiency. Static GNN models generated inconsistent predictions, forcing A* to explore more nodes than the Dijkstra baseline. A significant improvement was achieved only after the TGNN model was tasked with forecasting future speed rather than abstract cost, and this prediction was integrated into the classical geometric formula, preserving its mathematical stability while enriching it with knowledge about the future traffic state.

ISSN 1814-4225 (print)
ISSN 2663-2012 (online)

5. Results and Discussion

The obtained experimental results provide clear evidence of the impact that different heuristic design strategies have on the computational efficiency of the time-dependent A* algorithm. Table 2 and Figure 4 illustrate both the path quality (measured in travel minutes) and the search effort (measured in the number of explored nodes).

All proposed approaches succeeded in identifying the same optimal route with a travel time of 16.50 minutes, confirming that the integration of machine-learning-based heuristics does not compromise optimality. However, their search efficiency varied significantly depending on how well the heuristic captured temporal traffic dynamics.

The static models (LightGBM, GraphSAGE, and GATv2) provided limited improvements. While LightGBM was able to reduce the number of expanded nodes by 3.5 % compared to Dijkstra, its predictions remained inconsistent when the temporal context changed. Static GNN models (SAGE and GATv2), though able to encode graph topology, failed to adapt to traffic variability, producing "noisy" heuristics that occasionally misled the search and even required exploring more nodes than the uninformed baseline.

In contrast, the TGNN-based hybrid heuristic demonstrated the most robust performance. With two GCN layers, the model required exploring only 263 nodes, which is 41.8 % fewer than Dijkstra and 27.9 % fewer than the simple geometric heuristic. The addition of temporal modeling through the GRU component ensured smooth and consistent heuristic estimates across consecutive time intervals, significantly improving search convergence.

Furthermore, qualitative analysis revealed that the TGNN heuristic maintained stability under different traffic intensities, avoiding abrupt changes in predicted costs. This stability was achieved through the inclusion of the consistency loss term, which enforced the triangle inequality during training. As a result, the heuristic remained both admissible and smooth, enabling A* to follow an efficient, directed search trajectory toward the destination.

Overall, the experiments confirm that predicting a physical parameter (speed) rather than a final travel time (ETA) leads to more interpretable and stable heuristic functions. The TGNN-based approach thus bridges the gap between data-driven prediction and the theoretical guarantees required by classical search algorithms.

The proposed approach has several limitations. The method was validated on a single city graph (Irpin, 2,400 nodes), and its behavior on significantly larger networks requires further study. The TGNN model needs sufficient historical data for training. The hyperparameter λ in the combined loss function requires empirical tuning and may vary across topologies. Additionally, the current study uses a static graph topology with dynamic edge weights; changes in network structure (e.g., road closures) are not explicitly modeled.

Regarding computational cost, the TGNN inference adds minimal overhead: 0.44–0.65 ms per query vs. 0.83–1.01 ms for Dijkstra, confirming that execution time is lower than uninformed search. The model contains approximately 205K parameters (32% fewer than comparable architectures such as DCRNN), enabling efficient inference on consumer-grade hardware. Under drastic changes (accidents, missing data), the hybrid design preserves admissibility: even with inaccurate speed predictions, the classical formula bounds the estimate from below. Applying the model to a different city requires retraining on local traffic data, as the model learns city-specific spatial and temporal patterns; however, the training pipeline and architecture are fully general-purpose and do not depend on city-specific assumptions. For larger networks (e.g., New York), the GCN–GRU architecture scales with $O(|V|\times K)$ complexity; distributed frameworks would be needed for graphs exceeding tens of thousands of nodes.

6. Conclusions

This study proposed and experimentally validated a hybrid approach for constructing admissible and consistent heuristics for the **time-dependent A*** algorithm using **Temporal Graph Neural Networks (TGNNs)**.

The main findings are as follows:

1. Naive substitution of heuristics is ineffective. Machine learning models such as LightGBM, GraphSAGE, and GATv2, when used to directly predict total travel time (ETA), often violated consistency and produced unstable heuristics. This led to inefficient searches, with a higher number of expanded nodes compared to the classical A* baseline.

2. Predicting physical parameters improves heuristic quality. Reframing the problem – training TGNNs to predict future speed instead of direct ETA – produced a heuristic that remained both smooth and admissible. Incorporating temporal dependencies through the GRU layer allowed the heuristic to adapt to evolving traffic dynamics.

3. Hybrid TGNN-based heuristic significantly enhances A efficiency. The proposed model reduced the number of explored nodes by **41.8 % compared to Dijkstra** and by **27.9 % compared to the simple geometric heuristic**, while maintaining optimal path quality.

4. Consistency-aware training ensures stable search behavior. The inclusion of a consistency loss term

prevented violations of the triangle inequality, enabling the heuristic to remain theoretically valid and computationally efficient.

These results demonstrate that **combining classical search theory with data-driven prediction** can substantially improve routing efficiency in time-dependent networks.

Future work: Several directions for future research are identified:

- – Adaptive model retraining: automatic detection of performance degradation and dynamic retraining based on spatio-temporal data drift;

- – Multimodal feature integration: extending the TGNN model with additional parameters such as traffic density, weather, or road incidents;

- – Scalable deployment: developing distributed training and inference frameworks to ensure real-time performance on large-scale urban networks;

Contributions of authors: conceptualization, methodology, formulation of tasks, writing – review and editing – **Larysa Globa**; development of model, experiment, analysis, writing – original draft preparation – **Andrii Liashenko**.

Conflict of Interest

The authors declare that they have no conflict of interest in relation to this research, whether financial, personal, author ship or otherwise, that could affect the research and its results presented in this paper.

Financing

This research was conducted without financial support.

Data Availability

The manuscript has no associated data.

Use of Artificial Intelligence

The authors confirm that they did not use artificial intelligence technologies when creating the current work.

All authors have read and agreed to the published version of this manuscript.

References

1. *INRIX*. *INRIX Global Traffic Scorecar*d. INRIX, 2024. Available at: https://inrix.com/scorecard/#form-download-the-full-report (accessed 06 October 2025).

2. 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*, 1968, vol. 4, no. 2, pp. 100–107. DOI: 10.1109/TSSC.1968.300136.

3. Werner, N., & Zeitz, T. Combining predicted and live traffic with time-dependent A potentials. *Proc. European Symposium on Algorithms (ESA 2022). Leibniz International Proceedings in Informatics (LIPIcs)*, vol. 244. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2022, pp. 89:1–89:15. DOI: 10.4230/LIPIcs.ESA.2022.89.

4. Ron, D., Yusufzai, F. A., Kwayke, S., & Roy, S. Time-dependent network topology optimization for LEO satellite constellations. *IEEE Conference on Aerospace and Electronic Systems*, 2025. DOI: 10.48550/arXiv.2501.13280.

5. Zhang, Q., Shao, W., Shao, Z., & Pi, D. Graph-based reinforced multi-objective optimization for distributed heterogeneous flexible job shop scheduling problem under nonidentical time-of-use electricity pricing. *Expert Systems with Applications*, 2025. Available at: https://www.sciencedirect.com/science/article/abs/pii/S0957417425020470 (accessed 06 October 2025).

6. Maue, J., Sanders, P., & Matijevic, D. Goal-directed shortest-path queries using precomputed cluster distances. In: *Experimental Algorithms: 5th International Workshop, WEA 2006. Lecture Notes in Computer Science*, vol. 4007. Springer, Berlin, 2006, pp. 316–327. DOI: 10.1007/11764298_29. Available at: https://link.springer.com/chapter/10.1007/11764298_29 (accessed 06 October 2025).

7. Wu, F., & Wu, L. DeepETA: A spatial-temporal sequential neural network model for estimating time of arrival in package delivery system. In: *Proc. 30th International Joint Conference on Artificial Intelligence (IJCAI-21)*, 2021, pp. 2822–2828. DOI: 10.24963/ijcai.2021/388.

8. Chen, H., Huang, J., Lu, Y., & Huang, J. Multi-scale spatio-temporal graph neural network for urban traffic flow prediction (STGMS). *Scientific Reports*, 2025, vol. 15, no. 1, article 26732. DOI: 10.1038/s41598-025-11072-0.

9. Xiao, Y., Wang, J., Ding, Z., & Zhao, L. Adaptive spatio-temporal dynamic graph convolutional network (AST-DGCN) for urban traffic forecasting. *Scientific Reports*, 2025, vol. 15, no. 1, article 12261. DOI: 10.1038/s41598-025-12261-7.

10. Cooke, K. L., & Halsey, E. The shortest route through a network with time-dependent internodal transit times. *Journal of Mathematical Analysis and Applications*, 1966, vol. 14, no. 3, pp. 493–498. DOI: 10.1016/0022-247X(66)90009-6.

11. Orda, A., & Rom, R. Shortest-path and minimum-delay algorithms in networks with time-dependent edge-length. *Journal of the ACM*, 1990, vol. 37, no. 3, pp. 607–625. DOI: 10.1145/79147.214078.

12. Dreyfus, S. E. An appraisal of some shortest-path algorithms. *Operations Research*, 1969, vol. 17, no. 3, pp. 395–412. DOI: 10.1287/opre.17.3.395.

13. Zhao, L., Song, Y., Zhang, C., Liu, Y., Wang, P., Lin, T., & Deng, M. T-GCN: A temporal graph convolutional network for traffic prediction. *IEEE*

ISSN 1814-4225 (print)
ISSN 2663-2012 (online)

Transactions on Intelligent Transportation Systems, 2019. DOI: 10.1109/TITS.2019.2935152.

14. Haklay, M., & Weber, P. OpenStreetMap: User-generated street maps. *IEEE Pervasive Computing*, 2008, vol. 7, no. 4, pp. 12–18. DOI: 10.1109/MPRV.2008.80.

15. Boeing, G. OSMnx: New methods for acquiring, constructing, analyzing, and visualizing complex street networks. *Computers, Environment and Urban Systems*, 2017, vol. 65, pp. 126–139. DOI: 10.1016/j.compenvurbsys.2017.05.004.

16. Ke, G., Meng, Q., Finley, T., Wang, T., Chen, W., Ma, W., Ye, Q., & Liu, T.-Y. LightGBM: A highly efficient gradient boosting decision tree. In: *Advances in Neural Information Processing Systems 30 (NIPS 2017)*, pp. 3146–3154. Available at: https://proceedings.neurips.cc/paper/2017/hash/6449f44a102fde848669bdd9eb6b76fa-Abstract.html (accessed 06 October 2025).

17. Wu, Z., & et al. A comprehensive survey on graph neural networks. *IEEE Transactions on Neural Networks and Learning Systems*, 2020, vol. 32, no. 1, pp. 4–24.

18. Xu, D., Ruan, C., Korpeoglu, E., Kumar, S., & Achan, K. Temporal graph networks for deep learning on dynamic graphs. *arXiv preprint*, arXiv:2006.10637, 2020. DOI: 10.48550/arXiv.2006.10637.

19. Kipf, T. N., & Welling, M. Semi-supervised classification with graph convolutional networks. *arXiv preprint*, arXiv:1609.02907, 2016. DOI: 10.48550/arXiv.1609.02907.

20. Hamilton, W. L., Ying, R., & Leskovec, J. Inductive representation learning on large graphs. In: *Advances in Neural Information Processing Systems 30 (NeurIPS 2017)*, pp. 1024–1034. Available at: https://papers.nips.cc/paper/2017/hash/5dd9db5e033da9c6fb5ba83c7a7ebea9-Abstract.html (accessed 06 October 2025).

21. Veličković, P., Cucurull, G., Casanova, A., Romero, A., Liò, P., & Bengio, Y. Graph attention networks. *ICLR 2018*. DOI: 10.48550/arXiv.1710.10903.

22. Seo, Y., Defferrard, M., Vandergheynst, P., & Bresson, X. Structured sequence modeling with graph convolutional recurrent networks. In: *Neural Information Processing (ICONIP 2018)*. Lecture Notes in Computer Science, vol. 11301. *Springer*, 2018, pp. 362–373. DOI: 10.1007/978-3-030-04167-0_33.

23. Cho, K., van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., & Bengio, Y. Learning phrase representations using RNN encoder–decoder for statistical machine translation. *arXiv preprint*, arXiv:1406.1078, 2014. DOI: 10.48550/arXiv.1406.1078.

24. Oono, K., & Suzuki, T. Graph neural networks exponentially lose expressive power for node classification. In: International Conference on Learning Representations (ICLR 2020). DOI: 10.48550/arXiv.1905.10947.

25. Liu, Z., Shojaee, P., & Reddy, C. K. Graph-based multi-ODE neural networks for spatio-temporal traffic forecasting (GRAM-ODE). *arXiv preprint*, arXiv:2305.18687, 2023. DOI: 10.48550/arXiv.2305.18687.

26. Liu, Y., & Meidani, H. Fast graph neural network approximations for shortest distances and route estimation under uncertainty. *arXiv preprint*, arXiv:2501.09803, 2025. DOI: 10.48550/arXiv.2501.09803.

27. Francies, R., Zhu, K., & Tan, C. Continual learning-enhanced spatio-temporal graph neural networks for dynamic traffic forecasting. *Complex & Intelligent Systems*, 2025, vol. 11, article 20497. DOI: 10.1007/s40747-025-02049-7.

Received 07.09.2025, Received in revised form 25.12.2025
Accepted date 15.01.2026, Published date 22.01.2026

ПРИСКОРЕННЯ ПОШУКУ НА ОСНОВІ АЛГОРИТМУ A* У ЧАСОЗАЛЕЖНИХ ГРАФАХ ЗА ДОПОМОГОЮ НАВЧЕНИХ ЕВРИСТИК

А. В. Ляшенко, Л. С. Глоба

Сьогодні концепція Інтернету речей (IoT) охоплює все більше сфер сучасного життя, зокрема інтелектуальні транспортні системи. Ефективне керування трафіком в таких системах характеризується жорсткими вимогами до обчислювальної складності алгоритмів. Одним із складних завдань є пошук найкоротшого шляху в завантажених міських транспортних мережах, які представляють у вигляді графа з часозалежною вагою ребер. Для оцінки шляху, який залишився, традиційно використовуються статичні евристики, що базуються на геометричній відстані. Однак вони не враховують динаміку трафіку, що обмежує їхню ефективність. Сучасним напрямком вирішення цієї задачі є застосування методів машинного навчання для створення більш "розумних", адаптивних евристик. **Предметом дослідження** є методи підвищення ефективності алгоритму пошуку A* шляхом створення евристичних функцій на основі машинного навчання. **Метою роботи** є збільшення швидкодії, а саме зменшення кількості переборів вузлів, алгоритму A* в часозалежному графі за рахунок навчання евристичної функції. **Завданнями дослідження** є проведення порівняльного аналізу моделей машинного навчання, які можна використати в якості евристичної функції (LightGBM, GNN, TGNN); розробка гібридного методу, де TGNN прогнозує майбутній фізичний стан системи; та експериментальна оцінка виграшу в ефективності запропонованого підходу. **Отримано наступні результати.** Знайдене значення евристичної функції h в алгоритмі A* дозволяє зменшити кількість переборів

вузлів для знаходження найкоротшого шляху.. Модель була навчена на реальних GPS-даних, наданих українським ride-hailing провайдером, і протестована на дорожньому графі міста Ірпінь. Результати показали, що модель TGNN, яка вміє прогнозувати швидкість в момент часу $v_{TGNN}(u,t)$ для сусідніх вузлів, в яких міститься агрегована інформація від інших сусідніх вузлів досить добре себе показує у часозалежних графах. $h(u,t)$, евристика може скоротити обсяг пошуку (кількість досліджених вершин) на 28% порівняно з класичною евристикою при збереженні оптимальної якості маршруту. **Висновки.** Наукова новизна полягає у знаходженні такої моделі машинного навчання, що можна використати у якості евристичної функції, що вирішує протиріччя між високою середньою точністю ML-прогнозів та вимогами до консистентності A*. Доведено, що зміна задачі для ML-моделі з прогнозування загального часу на прогнозування фізичного параметра (швидкості) є ефективним способом підвищення продуктивності інтелектуальних систем пошуку шляху.

Ключові слова: A* алгоритм; часозалежний граф; машинне навчання; пошук найкоротшого шляху; часові графові нейронні мережі TGNN.

Ляшенко Андрій Володимирович – асп. каф. інформаційних технологій в телекомунікаціях, Інституту телекомунікаційних систем Національного технічного університету України «Київський політехнічний інститут імені Ігоря Сікорського», Київ, Україна.

Глоба Лариса Сергіївна – д-р техн. наук, проф., проф. каф. інформаційних технологій в телекомунікаціях, Інститут телекомунікаційних систем Національного технічного університету України «Київський політехнічний інститут імені Ігоря Сікорського», Київ, Україна.

Andrii Liashenko – PhD Student at the Department of Information Technologies in Telecommunications, Institute of Telecommunication Systems, National Technical University of Ukraine «Igor Sikorsky Kyiv Polytechnic Institute», Kyiv, Ukraine,
e-mail: andrey.lyashenko44@gmail.com, ORCID: 0009-0003-1714-414X, Scopus Author ID: 57559330900.

Larysa Globa – Doctor of Technical Sciences, Professor, Professor at the Department of Information Technologies in Telecommunications, Institute of Telecommunication Systems, National Technical University of Ukraine «Igor Sikorsky Kyiv Polytechnic Institute», Kyiv, Ukraine,
e-mail: lgloba@its.kpi.ua, ORCID: 0000-0003-3231-3012, Scopus Author ID: 57200923626.