Near-Future Traffic Forecasting for Planning-based Traffic Signal Optimisation

Mattia Chiari[1], **Francesco Percassi**[2], **Alfonso Emilio Gerevini**[1], **Mauro Vallati**[2]

[1]Dipartimento di Ingegneria dell'Informazione, Università degli Studi di Brescia, Italy
[2]School of Computing and Engineering, University of Huddersfield, United Kingdom
mattia.chiari@unibs.it, f.percassi@hud.ac.uk, alfonso.gerevini@unibs.it, m.vallati@hud.ac.uk

Abstract

Accurately forecasting traffic evolution is critical for effective urban traffic management, especially when integrated with operational control. While high-fidelity traffic simulators offer valuable insights, their high computational cost prevents daily use and real-time deployment. Data-driven approaches offer a computationally feasible alternative, at the cost of lower fidelity. This work bridges this gap by exploring the capabilities of near-future traffic forecasting tailored for traffic signal optimisation. We train data-driven surrogate models to predict the impact of various traffic signal configurations on network traffic over short horizons (up to 360 seconds). Our extensive evaluation on real-world data from a UK urban corridor demonstrates remarkable accuracy for multi-horizon predictions. Further, we demonstrate how these forecasts can be integrated into a planning-based traffic signal optimisation framework. We develop two novel, dedicated heuristics that leverage these predictions to guide the search. Our empirical results on real-world data demonstrate substantial improvements: the forecast-informed heuristics consistently improve solution quality while reducing the number of expanded nodes compared to state-of-the-art domain-independent heuristics.

Code — https://github.com/aiunibs/nf_traffic_forecasting

Introduction

The global urbanisation trend is leading to a rapid increase in the number of vehicles on urban roads, straining existing traffic management systems and exacerbating congestion and air quality issues (Zhang 2016). Existing control systems, such as SCOOT (Taale, Fransen, and Dibbits 1998), rely on reactive strategies that are effective at managing short-term traffic fluctuations but lack the foresight required to handle sudden demand peaks. Decentralised schedule-driven traffic signal control methods have also been proposed, such as SURTRAC (Smith et al. 2013).

Reactive adjustments may prove insufficient to handle rapid changes in demand, motivating the adoption of a planning-based module alongside SCOOT. The latter provides standard reactive control, while the planner is invoked when predicted traffic conditions suggest intervention.

Traffic simulators, driven by sensor data distributed across the target region, enable accurate exploration of complex traffic dynamics and their likely evolution. However, due to their computational complexity, their use is impractical for daily operations. To mitigate this issue, there is a growing interest in surrogate data-driven models (Cantatore et al. 2024), which have been explored for tasks ranging from traffic forecasting (Vlahogianni 2015), to serving as a substitute for traffic simulators (Fan et al. 2013; Katrakazas, Quddus, and Chen 2018). Such approaches, while computationally efficient, are less accurate and rely on low-level spatio-temporal granularity, providing a general overview of likely traffic evolutions, without directly supporting the operational decisions of traffic control operators (Gomes, Coelho, and Aidos 2023; Abirami et al. 2024).

This paper addresses the need for a computationally feasible yet accurate traffic forecasting methodology to provide actionable knowledge for a traffic signal optimisation agent. Our contributions are multifaceted. First, we introduce a framework to exploit near-future traffic forecasting for performing planning-based traffic signal optimisation. Second, we develop and train machine learning models to forecast the near-future traffic evolution of an urban network at the granularity of traffic signal settings. Third, we conduct an extensive evaluation of these models using real-world traffic data from a major urban corridor in the UK, demonstrating strong accuracy across multiple prediction horizons. Fourth, we design dedicated heuristic functions that leverage near-future forecasts and can be integrated into a planning-based traffic signal optimisation approach deployed in various regions of the UK (McCluskey and Vallati 2017; Kouaiti et al. 2024; Doria et al. 2026). This class of heuristics is particularly relevant in light of the growing interest in learned heuristics and value functions for general policies (Toyer et al. 2018; Shen, Trevizan, and Thiébaux 2020; Ferber et al. 2022; Ståhlberg, Bonet, and Geffner 2022; Rossetti et al. 2024; Chen and Thiébaux 2024; Chen, Trevizan, and Thiébaux 2024; Borelli et al. 2026b).

Our empirical results, validated with real-world data, show that the proposed agent based on forecast-informed heuristics yields superior solution quality while significantly reducing the computational burden of the search (i.e., fewer expanded nodes) compared with the currently deployed state-of-the-art domain-independent heuristics.

Background

In this section, we first describe the traffic signal optimisation problem and then discuss the characteristics of the target urban area and the planning-based approach designed to meet the deployment constraints.

Traffic signal control aims to determine a suitable green time for signals at controlled traffic junctions. Green times are grouped into coordinated stages, each representing a compatible set of traffic movements within a junction and corresponding green-lit signals. The order of stages is predetermined in a cycle. Legal and operational constraints limit the duration of stages and the cycle durations.

In essence, the traffic signal optimisation problem reduces to deciding, for each junction, how long each stage should remain active within the given constraints to ensure efficient traffic flow.

As a real-world case study, we focus on an urban corridor located in a mid-sized UK metropolitan area. This corridor spans approximately 1.3 km, includes six signalised junctions and 34 road links, and serves as a key route between a town centre, a major motorway, and nearby residential and commercial areas. Large events at the stadium and regular commuter flows often contribute to significant congestion along this route. Each junction operates between four and six stages and handles up to 17 distinct traffic movements.

In the target area, traditional traffic control systems, namely SCOOT (Taale, Fransen, and Dibbits 1998), are in operation. It is a reactive approach that adjusts signal timing at each cycle in response to real-time pressure sensor data.

The infrastructure operating in the target region, which is the most commonly used across the UK, required the design of dedicated automated planning approaches for ensuring deployability: the domain model known as FIRE (Fixed-Repetition) (Kouaiti et al. 2024), which relies on the hybrid planning formalism of PDDL+ (Fox and Long 2006) to jointly represent the discrete and continuous evolution of a traffic network. In the following, we formally define the relevant elements of FIRE and highlight that the planning system is being deployed in urban areas of the UK.

The network, composed of junctions and links, is represented as a directed graph $(\mathcal{J}, \mathcal{L})$, where $\mathcal{J}$ denotes the set of junctions and $\mathcal{L}$ the set of links, each corresponding to a road segment connecting two junctions. Each link has a fixed vehicle capacity and a time-varying occupancy that tracks the number of vehicles currently using the segment.

Each signalised junction operates according to a predefined signal cycle $\mathcal{S}_j$ composed of multiple *stages*, where each stage corresponds to a specific traffic phase that enables certain movements through the junction. Formally, each stage $st \in \mathcal{S}_j$ is associated with a set of permitted movements, defined as pairs of incoming and outgoing links (ℓ_{in}, ℓ_{out}), such that $\ell_{in} = (j_{in}, j)$ and $\ell_{out} = (j, j_{out})$, and with a *turn rate* that captures the expected vehicle flow for each movement when the stage is active. Turn rates are defined as positive rational values representing average macroscopic flows over a given time interval, derived from historical data. For convenience, we denote the set of all movements within the network by the set $\mathcal{M}$.

The automated planning control system operates by selecting *signal configurations* from a finite, predefined set per junction. Each configuration specifies how green time is distributed across the stages while respecting operational constraints such as stage order, intergreen times, and a fixed overall cycle duration. In this setting, *actions* take the form *changeConfiguration*(j, ϕ_1, ϕ_2) and encode the agent's decision to switch the active signal configuration at a junction j from ϕ_1 to an alternative one ϕ_2. The application of actions, by selecting which configurations are used over time, drives the exogenous evolution of the network, as modelled in PDDL+ via events and processes.

Processes are used to model continuous changes affecting numeric variables over time as long as a condition holds. In this context, the model includes one process for each permitted movement. These processes govern the evolution of link occupancies, with flow rates determined by the associated turn rate. A process activates when the currently active stage enables its movement, vehicles are present on the incoming link ℓ_{in}, and the outgoing link ℓ_{out} is not congested. Under this condition, the process continuously decreases the occupancy of ℓ_{in} and increases that of ℓ_{out}.

Events in PDDL+ are used to model instantaneous changes that occur as soon as specific conditions are satisfied. In this context, they capture the automatic transition between stages within a cycle.

In such a domain model, each link of interest ℓ is associated with a dedicated counter numeric variable, denoted by count_ℓ, which monotonically tracks the cumulative number of vehicles that have entered the link over time. The initial state is a full assignment of the numeric variables, providing a comprehensive description of the network in terms of link occupancies, available junction configurations, turn rates, and counters initialised to zero. The planning goal is formulated by requiring that, for each link of interest, specifically, those lying along the main corridor in our case study, the corresponding counter reaches a target threshold K_ℓ, i.e., $\langle \mathsf{count}_\ell \geq K_\ell \rangle$.

The corresponding PDDL+ planning task can be expressed as $\Pi = \langle V, I, G, A, E, P \rangle$ where V is the set of numeric and Boolean variables describing the network state (e.g., link occupancies, counters, and active configuration), I is the initial state assigning values to all variables in V, G is the goal set containing all conditions of the form $\langle \mathsf{count}_\ell \geq K_\ell \rangle$ for the relevant links, A is the set of controllable actions corresponding to configuration changes, and E and P are the sets of events and continuous processes representing the network's exogenous evolution, capturing stage transitions and vehicle flows along the links.

A signal plan for Π is defined as $\pi_t = \langle \pi, t_e \rangle$, where π is a timed sequence of change configuration actions and t_e is the makespan of the plan, i.e., the total duration required to achieve the goal. For a detailed formalisation of a general PDDL+ planning task Π and the definition of plan validity, which are beyond the scope of this work, we refer the reader to the work by Percassi, Scala, and Vallati (2025). For this paper, it is sufficient to regard Π as an action-based transition system describing the evolution of the traffic network.

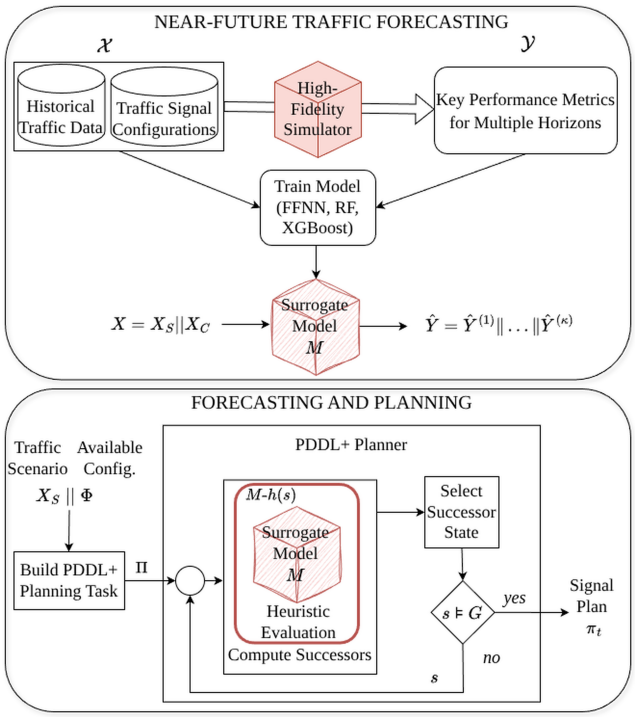

Figure 1: The proposed agent framework, combining Near-Future Traffic Forecasting (top), and Forecasting and Planning (bottom).

Framework Overview

Let us now turn our attention to the proposed architecture, shown in Figure 1, which integrates data-driven forecasting and automated planning to support traffic signal control in an operational setting. The architecture is divided into two main modules: *Near-Future Traffic Forecasting* (top), and *Forecasting and Planning* (bottom).

The first module (top) constructs a surrogate forecasting model that approximates a high-fidelity traffic simulator while being significantly less computationally expensive. To construct the surrogate model, we start by combining historical traffic data with a set of feasible signal control configurations. The configurations are defined to satisfy the network operational constraints (e.g., fixed cycle durations) and are systematically varied to cover a large portion of the admissible configuration space. Each combination of traffic conditions and control parameters in the training input set $\mathcal{X}$ is evaluated using a high-fidelity traffic simulator that reproduces the temporal evolution of the network over a finite horizon for the given scenario. At each time step, the simulator outputs the network state evolution, including link occupancies that represent the number of vehicles on each segment over time. From these trajectories, key performance metrics are derived at regular intervals, summarising the network evolution across multiple sub-horizons. These aggregated indicators constitute the target outputs $\mathcal{Y}$ used to train the forecasting model. Based on $\mathcal{X}$ and $\mathcal{Y}$, we train a surrogate model M approximating the traffic simulator.

Up to this point, our methodology follows the general structure of existing approaches, such as Cantatore et al. (2024). However, unlike these works, our surrogate model predicts network evolution given a specific control configuration, enabling its integration into a planning system that generates signal plans.

This integration forms the second module, illustrated in the lower panel of Figure 1, where M is embedded within a PDDL+ planning system.

In our approach, the traffic signal optimisation problem is encoded as a PDDL+ planning task Π, which is built based on a vector X_S describing the initial network condition and a predefined pool of signal control configurations Φ, as required by the FIRE model. The PDDL+ planner performs a forward search in the state space, starting from the initial state, and repeatedly applying actions to generate successor states. In this context, a heuristic function $h(s)$, combined with a search strategy, is used to prioritise state expansion until the goal condition holds ($s \models G$), yielding a signal plan π_t to be evaluated and deployed.

For each successor state s generated during search, the model M is queried to obtain short-horizon forecasts of network behaviour, which are then mapped into a scalar estimating the distance of s from the goal condition, yielding the data-driven heuristic M-$h(s)$. Because forward search may require thousands of expansions, the heuristic must be computationally efficient. Hence, the surrogate model M should favour fast querying over complexity, prioritising speed without sacrificing much accuracy. In our case, we adopt lightweight models for M: feed-forward neural networks (FFNN), random forests (RF), and gradient-boosted trees (XGBoost), as in Cantatore et al. (2024).

We are now in the best position to highlight two interesting properties of this architecture. First, the surrogate model is independent of the PDDL+ model in use, i.e., of the control formalisation adopted, provided that the latter exposes a full numeric description of the network state and active configuration for input to M. Second, the approach is extensible: the metric set can be expanded as needed (e.g., incremental link counters, average congestion, journey times on subnetworks, or the whole network). These metrics can be arbitrarily composed to craft the heuristic M-$h(s)$, steering the planner to produce signal plans that prioritise the operational objectives at hand. This is crucial, as current PDDL+ planning systems offer limited native support for optimising complex performance metrics (Percassi, Scala, and Vallati 2025); accordingly, we encode them in M-$h(s)$.

In the following sections, we detail the two modules and present corresponding experimental results. In this initial proof of concept, we evaluate the effectiveness of the agent by focusing on network throughput, measured via cumulative counters count_ℓ defined in the previous section.

Near-Future Traffic Forecasting

In this section, we introduce our surrogate model for near-future traffic forecasting. The aim is to predict key performance metrics for the target urban region at a set of future time intervals, assuming that the traffic signal configuration at each junction remains unchanged.

To create our model, we need to represent the *traffic condition* of the road network at a given instant, defined as $X = X_S \| X_C$, where X_S and X_C are, respectively, the *scenario* and *configuration vectors*. The scenario vector contains all historical traffic data, that is, information that cannot be directly controlled by the traffic infrastructure (e.g., SCOOT in our context): occupancies X_{occ} and turn rates X_{tr}. In particular, let $\mathcal{L} = \{l_1, \ldots, l_{|\mathcal{L}|}\}$ be the set of ordered and indexed links for the considered urban region. The occupancy vector is defined as $X_{\mathsf{occ}} = [x_{\mathsf{occ}}^{\ell_1}, \ldots, x_{\mathsf{occ}}^{\ell_{|\mathcal{L}|}}]$, where x_{occ}^{ℓ} represents the occupancy for the link ℓ. Similarly, let $\mathcal{M} = \{m_1, \ldots, m_{|\mathcal{M}|}\}$ be the ordered and indexed set of movements. The turn rate vector is defined as $X_{\mathsf{tr}} = [x_{\mathsf{tr}}^{m_1}, \ldots, x_{\mathsf{tr}}^{m_{|\mathcal{M}|}}]$ where x_{tr}^{m} represents the number of cars transited for movement m. The traffic infrastructure can affect the configuration vector, which specifies the traffic signal configurations adopted in the considered area at a given instant. Given the ordered set of junctions $\mathcal{J} = \{j_1, \ldots, j_{|\mathcal{J}|}\}$, the configuration vector is defined as $X_C = X_{j_1} \| \ldots \| X_{j_{|\mathcal{J}|}}$, where X_j collects the green times assigned to all the stages of junction j. In particular, given $\mathcal{S}_j = \{st_1, \ldots, st_{|\mathcal{S}_j|}\}$ as the set of stages associated to junction j, the green times of junction j are represented as $X_j = [x_{\mathsf{gt}}^{st_1}, \ldots, x_{\mathsf{gt}}^{st_{|\mathcal{S}_j|}}]$, where x_{gt}^{st} contains the green time for stage st.

The target vector Y that we want to predict represents the evolution of traffic in the near future, starting from the initial condition X. The target vector is defined as $Y = Y^{(1)} \| \ldots \| Y^{(\kappa)}$, where κ is the number of forecasting horizons considered, and each subvector $Y^{(i)}$ represents the change in the network state over the interval $i \cdot \Delta t$, with Δt denoting the granularity (in seconds) used for each prediction step. In general, the output vector can include any traffic-related metric of interest, such as congestion levels, journey times, or counters, depending on the context. Since we are evaluating the effect of the current junction configuration on traffic, we assume that the configuration vector X_C is used for the entire forecast horizon. To represent traffic evolution, we focus on counters. Therefore, each subvector is defined as $Y^{(i)} = [y_{\mathsf{count}}^{\ell_1(i)}, \ldots, y_{\mathsf{count}}^{\ell_{|\mathcal{L}|}(i)}]$, where each $y_{\mathsf{count}}^{\ell(i)}$ denotes the increase in the counter of link ℓ over a time step of $i \cdot \Delta$.

In the considered target region, the input vector has a total size $|X| = 137$, comprising $|X_{\mathsf{gt}}| = 20$, $|X_{\mathsf{occ}}| = 35$, and $|X_{\mathsf{tr}}| = 82$. For the output, we consider four prediction horizons ($\kappa = 4$), each spaced by a time window of $\Delta t = 90$ seconds. Given 12 links of interest, the resulting output vector has a dimensionality of $|Y| = 48$.

Dataset Generation

To construct the dataset $\mathcal{D}$ used for training our model, we collected historical traffic data from the target urban network. We focused on the period from January to April 2022, selecting data from weekday mornings (7:00–10:00 AM) and evenings (4:00–7:00 PM), which correspond to typical peak traffic hours.

From this data, we extracted 1,400 traffic scenarios, following the procedure introduced in Bhatnagar et al. (2022b), each represented by a vector X_S, which describes the state of the network at a given point in time. We denote the collection of all such vectors as $\mathcal{X}_S$. These scenarios serve a dual purpose: they are used to extract the network state at a specific point in time, represented by a vector X_S, and also serve as the basis for subsequent traffic simulations under various control configurations. To incorporate information on the signal control applied to each traffic condition, we generated synthetic signal control configurations for each historical scenario. Specifically, for each $X_S \in \mathcal{X}_S$, we created a set of 100 network configurations, denoted as $\mathcal{X}_C(X_S)$. Each of these represents an assignment of green time durations across all controllable junctions in the network. These synthetic configurations adhere to a fixed cycle duration of 90 seconds and are compliant with the area's constraints.

By combining each scenario vector with its corresponding configuration vectors, we built a set of input vectors $\mathcal{X} = \{X_S \| X_C \mid X_S \in \mathcal{X}_S,\ X_C \in \mathcal{X}_C(X_S)\}$.

For each $X \in \mathcal{X}$, we rely on the framework proposed by Bhatnagar et al. (2022a) to simulate the traffic dynamics over a 360-second horizon, keeping the assigned signal configuration fixed throughout the simulation. For each considered forecast horizon $i \cdot \Delta t$, we collect the values of relevant traffic counters. As shown by Bhatnagar et al. (2022a), this approach produces an accurate simulation within a 900-second horizon. In this way, each element of $\mathcal{X}$, which encodes both a traffic state and a signal control, is paired with its simulated evolution over time Y, forming a pair in the dataset $\mathcal{D}$.

We split the dataset $\mathcal{D}$ into two different subsets: $\mathcal{D}_{train}$, and $\mathcal{D}_{test}$. The dataset $\mathcal{D}_{test}$ contains 280 scenarios, each associated with its corresponding relative configuration vectors (i.e., $280 \times 100 = 28000$ tuples $\langle X, Y \rangle$). Please note that these scenario vectors are not included in $\mathcal{D}_{train}$ and are therefore not used during the training process. It is also worth noting that the configuration vectors in $\mathcal{D}_{test}$ may not have been seen during training, as the 100 configurations selected for each scenario are drawn from a large pool of synthetic configurations. We selected this test dataset to assess the generalisability of our model to unseen inputs. Finally, $\mathcal{D}_{train}$ contains all the remaining instances in $\mathcal{D}$.

Model Selection and Hyperparameter Tuning

Given the nature of the traffic forecasting task and the need for fast inference and low computational overhead, we focused on lightweight and efficient models. Specifically, we selected Feedforward Neural Networks (FFNNs) and two ensemble methods based on decision trees: Random Forests (RF) (Breiman 2001) and Extreme Gradient Boosting (XGBoost) (Chen and Guestrin 2016) as candidate models.

For each model, we select 100 hyperparameter configurations using Random Search and evaluate each configuration using Mean Squared Error (MSE) as the target metric. We use MSE for model selection and report MAE for interpretability. To assess the most suitable model and its hyperparameters for our task in a robust and unbiased manner, we employ 5-fold cross-validation on the dataset $\mathcal{D}_{train}$. To ensure that each fold is representative of the overall data

Model	Hyperparameter	Range
FFNN	batch_size:	{32, 64}
	hidden_layers:	{1, 10}
	neurons:	{100, 10000 }
RF	max_depth:	$\{null, 20\}$
	min_samples_leaf:	{1, 50}
	min_samples_split:	{2, 50}
	n_estimators:	[100, 1500]
XGBoost	learning_rate:	{0.05, 1}
	max_depth:	$\{null, 20\}$
	min_child_weight:	[1, 10]
	n_estimators:	[100, 1500]

Table 1: Hyperparameters and ranges for each model.

Model	MAE				
	Y	$Y^{(1)}$	$Y^{(2)}$	$Y^{(3)}$	$Y^{(4)}$
FFNN	11.1362	5.4989	9.4258	12.9955	16.6245
RF	4.8971	2.8423	4.3346	5.5757	6.8358
XGBoost	**0.7906**	**0.2779**	**0.6558**	**0.9843**	**1.2443**

Table 2: Results of the best model for each type obtained from the hyperparameter tuning process. In bold, we report the best results. Column Y reports the average MAE for the predicted counter increments among the 5 folds on dataset $\mathcal{D}_{train}$. Column $Y^{(i)}$ reports the average MAE for the counter predicted after $i \cdot \Delta t$ seconds, where $\Delta t = 90s$, among the 5 folds.

distribution, we construct a stratified partitioning of $\mathcal{D}_{train}$. Specifically, we stratify by scenario so that each fold contains a balanced number of configurations for each scenario. All model-training experiments were conducted on a machine equipped with an Intel Xeon Gold 6140 CPU running at 2.30 GHz, using Ubuntu 24.04 LTS. For each experiment, we allocated 16 GB of memory. Table 1 reports the tuned hyperparameters and the adopted range for each approach. For FFNNs, we use a linearly decreasing number of neurons across layers, from the selected initial hidden size to the output dimension. The number of neurons in each layer is computed proportionally, ensuring a smooth reduction in model capacity toward the output layer. We use the linear activation function for the output layer and the ReLU activation function for all the others. Moreover, we adopt the Adam (Kingma and Ba 2015) optimiser to train the model and use the Mean Squared Error (MSE) as both the loss function and the evaluation metric for RF and XGBoost.

Table 2 shows the results of the hyperparameter tuning process. For each model type (one for each row), we indicate the results obtained by the best-performing model in terms of Mean Absolute Error (MAE). The MAE value represents the average number of vehicles by which the model's predictions deviate from the simulated ground truth, thus providing a direct measure of prediction accuracy in terms of vehicle

Time	#	MAE				
		Y	$Y^{(1)}$	$Y^{(2)}$	$Y^{(3)}$	$Y^{(4)}$
07:00 - 08:00 AM	5300	1.51	0.64	1.25	1.81	2.35
08:00 - 09:00 AM	4300	1.20	0.49	0.98	1.45	1.89
09:00 - 10:00 AM	4500	1.18	0.49	0.97	1.42	1.82
04:00 - 05:00 PM	5400	1.17	0.47	0.96	1.42	1.84
05:00 - 06:00 PM	3800	1.23	0.48	1.01	1.48	1.95
06:00 - 07:00 PM	4700	1.19	0.49	0.98	1.43	1.85
All	28000	1.25	0.51	1.03	1.51	1.96

Table 3: Results of the model on the test set $\mathcal{D}_{test}$. Column $Y^{(i)}$ reports the MAE computed on the counter values predicted after $i \cdot \Delta t$ seconds, where $\Delta t = 90s$.

count. In terms of average performance, XGBoost outperforms the other models by almost an order of magnitude, achieving an average MAE of 0.7906. Across columns $Y^{(1)}$ to $Y^{(4)}$, all approaches show increasing error with the forecast horizon. Given the reported results, we select XGBoost as the model of choice and use the best-performing model's hyperparameters for training. XGBoost is configured with a learning rate of 0.05, 1,362 estimators, a maximum tree depth of 20, and a minimum child weight of 10.

Model Training and Results

After selecting the best model and its configuration through cross-validation, we performed a final training phase to obtain the model used for evaluation. To validate this final model, we constructed a validation set $\mathcal{D}_{val}$ by randomly selecting 100 morning instances (between 7:00 AM and 10:00 AM) and 100 evening instances (between 4:00 PM and 7:00 PM) from the training dataset $\mathcal{D}_{train}$. The remaining instances in $\mathcal{D}_{train}$, excluding those selected for validation, were used to train the model. The model was configured with the same hyperparameters as the best-performing model identified during the cross-validation-based hyperparameter tuning phase.

Finally, the model is evaluated on the previously introduced test set $\mathcal{D}_{test}$, which comprises 280 scenarios not included in $\mathcal{D}_{train}$, each with all associated configuration vectors. This dataset is used to assess the model's ability to generalise to new traffic conditions. The MAE results on $\mathcal{D}_{test}$ are reported in Table 3. As reported in row "All", the model achieves an overall MAE of 1.25 on unseen scenarios in the test set. Column Y shows no significant variation in the computed MAE across different time slots. The counter increments predicted after $90s$ (column $Y^{(1)}$) have the lowest error. This is expected, as the effects of a given configuration become more evident over longer time horizons. Across all time slots, the error in the predicted counter increments increases with the forecast horizon. In the 7:00-8:00 AM time slot, the model shows slightly worse performance than the others. This may be explained by varying traffic demand during this time slot, with lower demand from 7:00 to 7:15 AM

and higher demand later on. Finally, Figure 2 shows qualitative predictions of the model at each forecast horizon. The model exhibits consistent behaviour, with no evident outliers. For comparison, we note that other models explored during hyperparameter tuning exhibited less consistent behaviour.

Forecasting and Planning

Building on the agent architecture described previously, we now detail how forecasts generated by the surrogate model M are integrated within the PDDL+ planning system. Specifically, we define model-informed heuristics that use these forecasts to guide the search, prioritising the exploration of states predicted to be closer to the goal (Figure 1, "Heuristic Evaluation") (Bonet and Geffner 2001).

We assume a trained surrogate model M implementing a mapping $\hat{Y} = M(X)$, where X encodes the current traffic state and signal configuration, and $\hat{Y} = \hat{Y}^{(1)} \| \dots \| \hat{Y}^{(\kappa)}$ collects the predicted counter increments over multiple forecast horizons. Particularly, in terms of notation, for each link ℓ, we denote by $\hat{y}^{\ell(i)}_{\mathsf{count}}$ the predicted increase of its cumulative counter at horizon $i \cdot \Delta t$.

We now present two strategies for integrating $\hat{Y}$ into heuristic evaluation. The first considers only a single horizon, while the second incorporates an arbitrary subset of multiple horizons. These two formulations differ in the type of estimated quantity and in the temporal information they leverage. The single-horizon heuristic evaluates the number of vehicles still required to reach the goal (*counter-to-goal*), whereas the multi-horizon variant estimates the time needed to do so (*time-to-goal*). This comparison also sheds light on whether incorporating multiple forecast horizons yields more effective guidance for search.

We denote a heuristic instantiated with model predictions as $M\text{-}h^{Z}_{\mathcal{I}}$, where M indicates the surrogate model (FFNN, RF, or XGBoost) used to generate the forecast vector, Z specifies the heuristic formulation, and $\mathcal{I} \subseteq \{1, \dots, \kappa\}$ is a non-empty subset of horizons considered when computing the heuristic.

Single-Forecast Counter-to-goal Heuristic

For the considered representation, the goal of the PDDL+ planning task is to move at least K_ℓ vehicles across each link ℓ in the goal. The most direct way to integrate model forecasts into a heuristic is therefore to measure, for each goal link, the difference between the required threshold K_ℓ and the predicted future counter value at a given forecast horizon t_i. We denote this heuristic strategy as "Δcount".

When more links are specified in the goal, the individual estimates can be aggregated by summing them across all relevant links. This gives rise to heuristic $M\text{-}h^{\Delta\mathsf{count}}_{\mathcal{I}}$. Moreover, since this heuristic considers only a single forecast horizon $i \cdot \Delta t$, the set $\mathcal{I}$ is defined as the singleton $\{i\}$.

Formally, let s be a state encountered during search on a FIRE planning task, X the feature vector extracted from s. Here, we denote by $s[\mathsf{count}_\ell]$ the current value of the counter associated with link ℓ in state s.

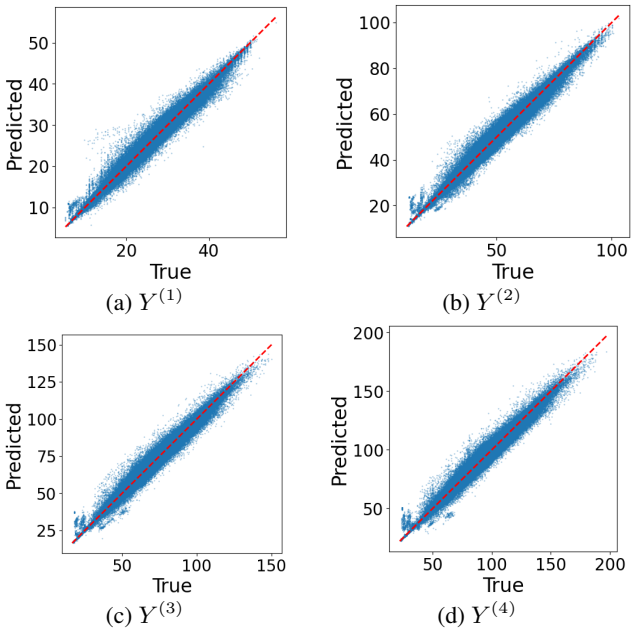

Figure 2: Results of the model on $\mathcal{D}_{test}$. Plot $Y^{(i)}$ illustrates the results for forecast horizon $i \cdot \Delta t$. For each plot on the x axes, we report the actual counter values, while on the y axes we report the predicted ones.

Heuristic $M\text{-}h^{\Delta\mathsf{count}}_{\{i\}}$ is defined as:

$$M\text{-}h^{\Delta\mathsf{count}}_{\{i\}}(s) = \sum_{\langle \mathsf{count}_\ell \geq K_\ell \rangle \in G} \max(K_\ell - s[\mathsf{count}_\ell] - \hat{y}^{\ell(i)}_{\mathsf{count}}, 0).$$

The summation iterates over all goal conditions of the form $\langle \mathsf{count}_\ell \geq K_\ell \rangle$ in G, corresponding to the links that define the target corridor. For each such link, the heuristic computes the number of vehicles still required to meet the threshold K_ℓ, adjusted by the predicted counter increment $\hat{y}^{\ell(i)}_{\mathsf{count}}$. The $\max(\cdot, 0)$ operator ensures non-negative cost estimates, returning zero when the goal is (predicted to be) satisfied at horizon $i \cdot \Delta t$.

Multiple-Forecast Time-to-goal Heuristic

The Δcount heuristic described above leverages only a single forecast horizon. However, since model M produces forecasts over multiple horizons, it is natural to design a heuristic that natively leverages several horizons.

We therefore introduce a second heuristic, denoted "LS" (least squares), which aggregates multiple predictions by interpolating them over time. For each link ℓ, we consider its predicted counter increments across multiple horizons $\mathcal{I}$ to approximate its short-term evolution. Given the set of points

$$\{(0, 0)\} \cup \{(t_i, \hat{y}^{\ell(i)}_{\mathsf{count}}) \mid i \in \mathcal{I}\},$$

where $\hat{y}^{\ell(i)}_{\mathsf{count}}$ denotes the predicted increase in the counter for link ℓ at horizon $t_i = i \cdot \Delta t$, we fit a line $y_\ell(t) = m_\ell t + b_\ell$ via least squares interpolation.

Once the parameters (m_ℓ, b_ℓ) are obtained, we estimate the time t^*_ℓ at which the interpolated counter trajectory reaches the target threshold K_ℓ. Formally:

$$t^*_\ell = \begin{cases} \max\left(\frac{K_\ell - s[\mathsf{count}_\ell] - b_\ell}{m_\ell}, 0\right) & \text{if } m_\ell > 0 \\ +\infty & \text{otherwise} \end{cases}$$

If the interpolated trend is increasing ($m_\ell > 0$), the intersection time corresponds to the earliest point where $y_\ell(t) = K_\ell$. (Again, we use $\max(\cdot, 0)$ to ensure non-negative values when the condition is already satisfied.) Otherwise, if the trend is non-increasing ($m_\ell \leq 0$), the goal is predicted to be unreachable within the forecast horizon. (Note that, consistently with the model formulation and the training data, m_ℓ is expected to be positive.)

Finally, we can then define the heuristic by simply summing these contributions over all goal conditions:

$$M\text{-}h^{\text{LS}}_{\mathcal{I}}(s) = \sum_{\langle \mathsf{count}_\ell \geq K_\ell \rangle \in G} t^*_\ell.$$

Note that, since the point $(0, 0)$ is included in the interpolation set, the method remains applicable even when $\mathcal{I}$ is a singleton.

Experimental Evaluation

In this section, we experimentally evaluate heuristics derived from forecasting models and assess their effectiveness in guiding the search for traffic signal plans within the proposed agent framework. Following the comparative evaluation of the models presented earlier, we selected XGB as the basis for constructing the heuristic estimators. Specifically, we considered two heuristics:

- XGB-$h^{\Delta\text{count}}_{\mathcal{I}}$, for $\mathcal{I} \in \{\{1\}, \{2\}, \{3\}, \{4\}\}$, corresponding to the Δcount strategy that uses model predictions at a single forecast horizon of 90, 180, 270, or 360 seconds;
- XGB-$h^{\text{LS}}_{\mathcal{I}}$, for $\mathcal{I} \in \{\{1\}, \{1, 2\}, \{1, 2, 3\}, \{1, 2, 3, 4\}\}$, corresponding to the LS strategy that aggregates multiple predicted horizons through linear interpolation.

As a baseline heuristic, we considered $h^{\max}$, a domain-independent heuristic provided by the ENHSP planner (Scala et al. 2020). This heuristic has been employed in prior work on PDDL+-based traffic signal optimisation (Kouaiti et al. 2024), which showed stable performance compared to alternative heuristics and competitive results against SCOOT. All forecasting-based heuristics proposed in this work were implemented within the same planning framework using the XGBoost4J API. All heuristics, including $h^{\max}$, were used with Greedy Best-First Search (GBFS), which ranks states based on the heuristic estimate.

Experiments were run on a machine with an Intel Xeon Gold 6140M CPU (2.30 GHz) running CentOS Linux 7 (Core). Each run had a 600-second time limit and 8 GB of memory. The implementation was run with Java 17.0.5.

As benchmarks, we considered the 280 planning tasks used to construct the testing set, namely, a collection of traffic scenarios that were never seen during model training. For these planning tasks, six alternative configurations were assigned to each junction, extracted from historical data of the configurations employed by SCOOT during the period of interest. These configurations are referenced in Kouaiti et al. (2024) as globally historical configurations (GHIST). Plan metrics are computed following the methodology described by Bhatnagar et al. (2022a). Generated plans are validated and replayed using the PPS validator (Scala, Percassi, and Vallati 2026) and a PDDL+ simulation model calibrated against SCOOT sensor data, which provides reliable estimates up to approximately 900 seconds.

Table 4 summarises the results averaged over the set of commonly solved problems. The reported metrics are averages computed across all planning tasks and include the plan makespan, the number of configuration changes, the number

GBFS+h	MS	$\lvert\pi\rvert$	EN	Time	Thpt
XGB-$h^{\text{LS}}_{\{1,2,3,4\}}$	**1489.02**	10.93	**5186.11**	**143.94**	**1310.28**
XGB-$h^{\text{LS}}_{\{1,2,3\}}$	1497.36	10.95	5398.50	146.67	1306.41
XGB-$h^{\text{LS}}_{\{1,2\}}$	1498.95	11.69	6994.88	192.60	1304.89
XGB-$h^{\text{LS}}_{\{1\}}$	1505.45	11.78	7575.33	211.20	1307.65
XGB-$h^{\Delta\text{count}}_{\{4\}}$	**1495.39**	9.69	2820.81	82.02	1303.41
XGB-$h^{\Delta\text{count}}_{\{3\}}$	1510.34	9.14	2507.45	72.75	1303.38
XGB-$h^{\Delta\text{count}}_{\{2\}}$	1511.07	9.43	2281.98	67.16	**1303.70**
XGB-$h^{\Delta\text{count}}_{\{1\}}$	1521.62	7.78	**1858.25**	**56.92**	1300.74
$h^{\max}$	1546.17	7.92	13663.04	**7.71**	1276.46

Table 4: Summary results. For each GBFS+h, we report the plan makespan (MS), the number of configuration changes ($|\pi|$), the number of expanded nodes (EN), the planning time in seconds (Time), and the throughput delivered by the network after 900 seconds (Thpt), all expressed as averages. The best results within each approach for $h^{\Delta\text{count}}$ and h^{LS} are highlighted in bold. The overall best results across all approaches are indicated in bold and underlined.

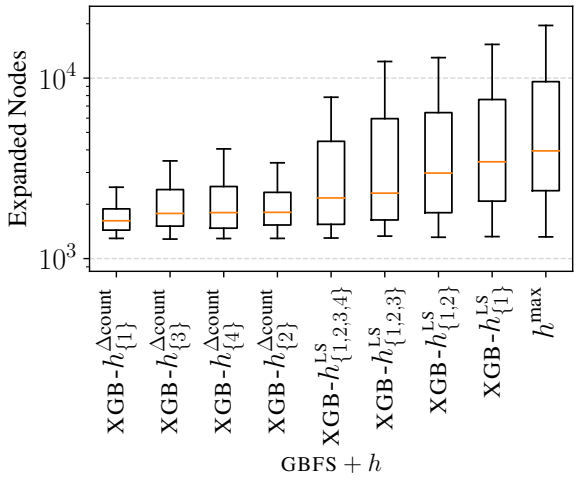

Figure 3: Boxplot of expanded nodes per heuristic approach, ordered by median.

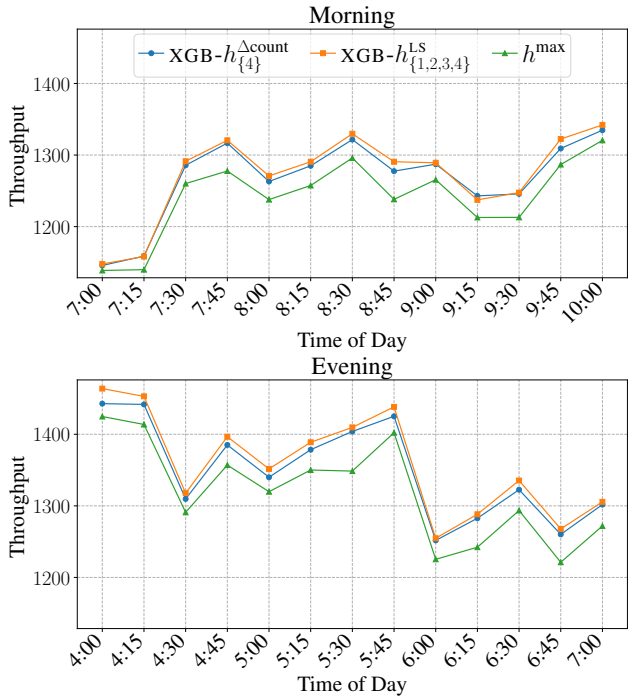

Figure 4: Average throughput after 900 seconds per time slot, computed by averaging the results of all planning tasks referring to the same time slot.

of expanded nodes, the planning time (in seconds), and the throughput after 900 seconds. The makespan denotes the total duration of the generated plan, reflecting how quickly the signal plan can move a specified number of vehicles through the corridor. The number of configuration changes quantifies the extent of control applied to the network. The number of expanded nodes serves as a proxy for heuristic informativeness, with fewer node expansions suggesting that the heuristic provides more effective search guidance. Finally, since the simulation environment remains reliable within time windows of up to 900 seconds, we included the network throughput measured at that point as a UTC-related performance measure.

Before discussing the results, we note that this study is intended as a proof of concept. The primary objective is to demonstrate the feasibility and potential of developing a traffic control agent that integrates heuristics derived from forecasting models into a planning framework for traffic signal optimisation, rather than producing fully optimised implementations. Therefore, runtime performance is not the focus of the analysis or of the optimisation. In the conclusions, we will outline potential technical enhancements to improve runtime performance.

Table 4 highlights several key trends. First, all XGB-based heuristics improve the makespan to varying degrees, depending on the heuristic strategy and forecast horizon considered. This suggests that, in principle, such heuristics can guide the search towards high-quality signal plans. Second, these heuristics significantly reduce the number of expanded nodes required to reach the goal, demonstrating, unsurprisingly, that they are more informed than the domain-independent heuristic. Although the XGB-based heuristics provide better guidance by narrowing the search space, this comes at the cost of increased planning time. While the selected models prioritise efficiency, their repeated evaluation across thousands of search nodes introduces a non-negligible overhead. However, the achieved performance still supports operational use.

Figure 3, which presents the box plots of expanded nodes across a population of commonly solved instances, complements the results reported in Table 4. Specifically, it can be observed that, in general, the LS strategy appears to be less informative than Δcount. Nevertheless, the number of expanded nodes depends on the forecast horizon considered.

For the Δcount strategy, the average number of expanded nodes tends to increase as the forecast horizon extends; correspondingly, the average makespan also improves. Conversely, for the LS strategy, incrementally incorporating more predictions reduces both the number of expanded nodes and the resulting makespan.

To conclude, we investigate a key metric in the UTC context: network throughput, focusing on the corridor of interest and comparing the best configurations for the two approaches. Specifically, we consider XGB-$h^{\Delta\text{count}}_{\{4\}}$ and XGB-$h^{\text{LS}}_{\{1,2,3,4\}}$. Figure 4 reports throughput measured at 900 seconds, averaged over instances within each time slot, for the selected approaches and the baseline. Notably, both approaches consistently improve the baseline, with gains varying across time slots. Interestingly, XGB-$h^{\Delta\text{count}}_{\{4\}}$, despite relying on a single horizon, achieves performance only slightly below that of XGB-$h^{\text{LS}}_{\{1,2,3,4\}}$ while expanding just over half as many nodes.

Conclusion and Future Work

In this paper, we present a framework for traffic signal optimisation that leverages near-future traffic forecasting models. Building on this, we developed forecasting models across multiple temporal horizons up to 360 seconds. We empirically demonstrated using data from a real-world network that tree-based ensemble models achieved higher accuracy, with XGBoost maintaining low prediction error across all forecast horizons. We then introduced an approach to integrate forecasts into a planning-based traffic signal optimisation framework deployable in the UK, by designing two novel heuristics that exploit predicted information to guide the search process. The experimental evaluation demonstrated that the proposed heuristics effectively guide search and produce high-quality plans.

Future work will focus on exploring a tighter integration of forecast evaluations and the underlying planning system, for instance, by computing predictions in batches whenever a node is expanded (Borelli et al. 2026a). An alternative is to use lighter-weight models, such as Ridge Regression (McDonald 2009). We are also interested in exploring multiple forecasting outputs and in investigating how they can be incorporated in the planning framework to support multi-objective optimisation.

Acknowledgments

Francesco Percassi and Mauro Vallati were supported by a UKRI Future Leaders Fellowship [grant number MR/Z00005X/1]. Mattia Chiari and Alfonso E. Gerevini were partially supported by project SERICS (PE00000014) under the MUR National Recovery and Resilience Plan funded by the European Union - NextGenerationEU, by project FAIR (B53C22003980006) under the MUR National Recovery and Resilience Plan funded by the European Union - NextGenerationEU, and by MUR PRIN-2020 project RIPER (n. 20203FFYLK).

References

Abirami, S.; Pethuraj, M.; Uthayakumar, M.; and Chitra, P. 2024. A Systematic Survey on Big Data and Artificial Intelligence Algorithms for Intelligent Transportation System. *Case Stud. Transp. Policy*, 17: 101247.

Bhatnagar, S.; Guo, R.; McCabe, K.; McCluskey, T. L.; Scala, E.; and Vallati, M. 2022a. Leveraging Artificial Intelligence for Simulating Traffic Signal Strategies. In *IEEE ITSC*, 607–612.

Bhatnagar, S.; Mund, S.; Scala, E.; McCabe, K.; McCluskey, T. L.; and Vallati, M. 2022b. On-the-Fly Knowledge Acquisition for Automated Planning Applications: Challenges and Lessons Learnt. In *ICAART*, 387–397.

Bonet, B.; and Geffner, H. 2001. Planning as heuristic search. *Artif. Intell.*, 129(1-2): 5–33.

Borelli, V.; Gerevini, A. E.; Scala, E.; and Serina, I. 2026a. LeapNP: A Modular Python Framework for Benchmarking Learned Heuristics in Numeric Planning. *Future Internet*, 18(2): 93.

Borelli, V.; Gerevini, A. E.; Scala, E.; and Serina, I. 2026b. Learning Heuristic Functions with Graph Neural Networks for Numeric Planning. In *AAAI*, 36172–36179.

Breiman, L. 2001. Random Forests. *Mach. Learn.*, 45(1): 5–32.

Cantatore, F.; Raimondi, G.; Oneto, L.; Coraddu, A.; Pasquale, C.; Siri, E.; Siri, S.; Sacone, S.; and Anguita, D. 2024. Traffic Simulator AI-Based Surrogate for an Urban Road Network. In *IEEE ITSC*, 116–123.

Chen, D. Z.; and Thiébaux, S. 2024. Graph Learning for Numeric Planning. In *NeurIPS*.

Chen, D. Z.; Trevizan, F. W.; and Thiébaux, S. 2024. Return to Tradition: Learning Reliable Heuristics with Classical Machine Learning. In *ICAPS*, 68–76.

Chen, T.; and Guestrin, C. 2016. XGBoost: A Scalable Tree Boosting System. In *KDD*, 785–794.

Doria, F.; Percassi, F.; Maratea, M.; and Vallati, M. 2026. A Domain-specific Heuristic for PDDL+-based Traffic Signal Optimisation. In *AAAI*, 36207–36216.

Fan, R.; Yu, H.; Liu, P.; and Wang, W. 2013. Using VISSIM simulation model and Surrogate Safety Assessment Model for estimating field measured traffic conflicts at freeway merge areas. *IET Intell. Transp. Syst.*, 7: 68–77.

Ferber, P.; Geißer, F.; Trevizan, F. W.; Helmert, M.; and Hoffmann, J. 2022. Neural Network Heuristic Functions for Classical Planning: Bootstrapping and Comparison to Other Methods. In *ICAPS*, 583–587.

Fox, M.; and Long, D. 2006. Modelling Mixed Discrete-Continuous Domains for Planning. *J. Artif. Intell. Res.*, 27: 235–297.

Gomes, B.; Coelho, J.; and Aidos, H. 2023. A survey on traffic flow prediction and classification. *Intell. Syst. Appl.*, 20: 200268.

Katrakazas, C.; Quddus, M.; and Chen, W. 2018. A Simulation Study of Predicting Real-Time Conflict-Prone Traffic Conditions. *IEEE Trans. Intell. Transp. Syst.*, 19(10): 3196–3207.

Kingma, D. P.; and Ba, J. 2015. Adam: A Method for Stochastic Optimization. In Bengio, Y.; and LeCun, Y., eds., *ICLR*.

Kouaiti, A. E.; Percassi, F.; Saetti, A.; McCluskey, T. L.; and Vallati, M. 2024. PDDL+ Models for Deployable yet Effective Traffic Signal Optimisation. In *ICAPS*, 168–177.

McCluskey, T. L.; and Vallati, M. 2017. Embedding Automated Planning within Urban Traffic Management Operations. In *ICAPS*, 391–399.

McDonald, G. C. 2009. Ridge Regression. *Wiley Interdiscip. Rev. Comput. Stat.*, 1: 93–100.

Percassi, F.; Scala, E.; and Vallati, M. 2025. On the Notion of Plan Quality for PDDL+. In *ICAPS*, volume 35, 102–111.

Rossetti, N.; Tummolo, M.; Gerevini, A. E.; Putelli, L.; Serina, I.; Chiari, M.; and Olivato, M. 2024. Learning General Policies for Planning through GPT Models. In *ICAPS*, 500–508.

Scala, E.; Haslum, P.; Thiébaux, S.; and Ramirez, M. 2020. Subgoaling Techniques for Satisficing and Optimal Numeric Planning. *J. Artif. Intell. Res.*, 68: 691–752.

Scala, E.; Percassi, F.; and Vallati, M. 2026. PPS: An Efficient Java-based Simulator for Time-Discrete PDDL+. In *AAAI (Demo Track)*, 41682–41684.

Shen, W.; Trevizan, F. W.; and Thiébaux, S. 2020. Learning Domain-Independent Planning Heuristics with Hypergraph Networks. In *ICAPS*, 574–584.

Smith, S.; Barlow, G.; Xie, X.-F.; and Rubinstein, Z. 2013. Smart Urban Signal Networks: Initial Application of the SURTRAC Adaptive Traffic Signal Control System. In *ICAPS*, volume 23, 434–442.

Ståhlberg, S.; Bonet, B.; and Geffner, H. 2022. Learning Generalized Policies without Supervision Using GNNs. In *KR*, 474–483.

Taale, H.; Fransen, W.; and Dibbits, J. 1998. The second assessment of the SCOOT system in Nijmegen. In *RTIC*, 109–113.

Toyer, S.; Trevizan, F.; Thiébaux, S.; and Xie, L. 2018. Action Schema Networks: Generalised Policies with Deep Learning. In *AAAI*, 6294–6301.

Vlahogianni, E. I. 2015. Optimization of traffic forecasting: Intelligent surrogate modeling. *Transp. Res. Part C: Emerg. Technol.*, 55: 14–23.

Zhang, X. Q. 2016. The trends, promises and challenges of urbanisation in the world. *Habitat Int.*, 54: 241–252.