Optimising Dynamic Traffic Distribution for Urban Networks with ASP

TPLP **24** (4): 825–843, 2024.

doi:10.1017/S1471068424000309 First published online 28 October 2024

Optimising Dynamic Traffic Distribution for Urban Networks with Answer Set Programming[*]

MATTEO CARDELLINI
University of Genova, Italy and Politecnico of Turin, Italy,
(e-mail: matteo.cardellini@edu.unige.it)

CARMINE DODARO and MARCO MARATEA
University of Calabria, Italy,
(e-mails: carmine.dodaro@unical.it, marco.maratea@unical.it)

MAURO VALLATI
University of Huddersfield, UK,
(e-mail: m.vallati@hud.ac.uk)

submitted 14 August 2024; accepted 13 September 2024

Abstract

Answer set programming (ASP) has demonstrated its potential as an effective tool for concisely representing and reasoning about real-world problems. In this paper, we present an application in which ASP has been successfully used in the context of dynamic traffic distribution for urban networks, within a more general framework devised for solving such a real-world problem. In particular, ASP has been employed for the computation of the "optimal" routes for all the vehicles in the network. We also provide an empirical analysis of the performance of the whole framework, and of its part in which ASP is employed, on two European urban areas, which shows the viability of the framework and the contribution ASP can give.

Keywords: answer set programming, optimization problems, traffic distribution

1 Introduction

Avoiding congestion and controlling traffic in urban scenarios is becoming nowadays of utmost importance due to the rapid growth of our cities' population and vehicles. The effective control of urban traffic as a mean to mitigate congestion can be beneficial in an economic, environmental and health way. At the end of the 21st century, the

[*] Carmine Dodaro and Marco Maratea were supported by Italian Ministry of Research (MUR) under PNRR project FAIR "Future AI Research", CUP H23C22000860006. Carmine Dodaro was supported by Italian Ministry of Research (MUR) under PNRR project Tech4You "Technologies for climate change adaptation and quality of life improvement", CUP H23C22000370006; and by GNCS-INdAM. Mauro Vallati was supported by a UKRI Future Leaders Fellowship [grant number MR/T041196/1].

world population is expected to increase to 10.9 Billion, adding well over 3 Billion people to the current population (Max Roser and Ortiz-Ospina 2013). This massive growth, which will directly translate in more vehicles roaming the streets of our cities, demands improvements in the transport infrastructure and a better utilisation of our roads for the purpose of avoiding congesting the network. Traffic jams have a negative impact on safety and fuel consumption, which directly translates to a higher cost for drivers and health issues for residents near highly trafficked roads, caused by bad air quality and noise pollution (Van Mierlo *et al*. 2004). Artificial Intelligence techniques, based on automated planning, have already been employed for optimising traffic flow, and more general in transportation (see, e.g., (Cenamor *et al*. 2014; Chrpa *et al*. 2016; Vallati *et al*. 2016; Ramírez *et al*. 2018; Cardellini *et al*. 2021; El Kouaiti *et al*. 2024)), with some benefits, but they fail to scale in the presence of large number of vehicles.

In this paper, we present an application in which answer set programming (ASP) (Gelfond and Lifschitz 1991; Niemelä 1999; Baral 2003; Brewka *et al*. 2011) has been successfully used in the context of dynamic traffic distribution for urban networks, within a more general framework devised for solving such a real-world problem. The framework, which allows to efficiently optimise and simulate traffic flow in a large roads' network with hundreds of vehicles, is composed of four phases: network analysis, domain-independent search, route optimisation, and mobility simulation. Within the framework, ASP is employed for representing and reasoning about the optimisation of the flow of traffic inside a road network by finding the best combination (schedule) of routes for all the vehicles in the network. We have performed an analysis on real-world traffic data from two European urban areas in UK and Italy, utilising the state-of-the-art Urban Mobility Simulator SUMO (López *et al*. 2018) to keep track of the state of the network: the analysis tested the correctness of the solution, and proved the efficiency and capabilities of the presented solution to reduce the metrics considered, sometimes significantly. Moreover, it shows the contribution ASP gives in terms of performance and metrics: all instances of Milton Keynes and Bologna up to 600 vehicles inside the network are solved, optimally, in a short time.

The paper is structured as follows. Section 2 presents preliminaries about ASP. Then, Section 3 introduces the problem and the solution framework. The last two phases of the framework, that is, the optimisation via ASP and the mobility simulator, are presented in Section 4 and Section 5, respectively, together with the experiments we performed on real data. The paper ends in Section 6 and 7 by discussing related work and by drawing some conclusions, respectively.

2 Background on ASP

ASP is a programming paradigm developed in the field of non-monotonic reasoning and logic programming. In this section, we overview the language of ASP. More detailed descriptions and a more formal account of ASP, including the features of the language employed in this paper, can be found in Brewka *et al*. (2011); Calimeri *et al*. (2020). Hereafter, we assume the reader is familiar with logic programming conventions.

https://doi.org/10.1017/S1471068424000309 Published online by Cambridge University Press

2.1 Syntax

Variables are strings starting with an uppercase letter, and constants are non-negative integers or strings starting with lowercase letters. A *term* is either a variable or a constant. A *standard atom* is an expression $p(t_1, \ldots, t_n)$, where p is a *predicate* of arity n and $t_1, \ldots, t_n$ are terms. An atom $p(t_1, \ldots, t_n)$ is ground if $t_1, \ldots, t_n$ are constants. A *ground set* is a set of pairs of the form $\langle consts{:}conj \rangle$, where *consts* is a list of constants and *conj* is a conjunction of ground standard atoms. A *symbolic set* is a set specified syntactically as $\{Terms_1 : Conj_1; \cdots ; Terms_t : Conj_t\}$, where $t > 0$, and for all $i \in [1, t]$, each $Terms_i$ is a list of terms such that $|Terms_i| = k > 0$, and each $Conj_i$ is a conjunction of standard atoms. A *set term* is either a symbolic set or a ground set. Intuitively, a set term $\{X{:}a(X, c), p(X); Y{:}b(Y, m)\}$ stands for the union of two sets: the first one contains the X-values making the conjunction $a(X, c), p(X)$ true, and the second one contains the Y-values making the conjunction $b(Y, m)$ true. An *aggregate function* is of the form $f(S)$, where S is a set term, and $f \in \{\#count, \#sum\}$ is an *aggregate function symbol*. An *aggregate atom* is of the form $f(S) \prec T$, where $f(S)$ is an aggregate function, $\prec \in \{<, \leq, >, \geq, \neq, =\}$ is an operator, and T is a term called guard. An aggregate atom $f(S) \prec T$ is ground if T is a constant and S is a ground set. An *atom* is either a standard atom or an aggregate atom. A *rule* r has the following form:

$$a_1 \mid \ldots \mid a_n :- b_1, \ldots, b_k, not\ b_{k+1}, \ldots, not\ b_m.$$

where $a_1, \ldots, a_n$ are standard atoms, $b_1, \ldots, b_k$ are atoms, $b_{k+1}, \ldots, b_m$ are standard atoms, and $n, k, m \geq 0$. A literal is either a standard atom a or its negation *not* a. The disjunction $a_1 \mid \ldots \mid a_n$ is the *head* of r, while the conjunction $b_1, \ldots, b_k, not\ b_{k+1}, \ldots, not\ b_m$ is its *body*. Rules with empty body and with only one atom in the head (i.e., $n = 1$) are called *facts*. Rules with empty head are called *constraints*. A variable that appears uniquely in set terms of a rule r is said to be *local* in r, otherwise it is a *global* variable of r. An ASP program is a set of *safe* rules, where a rule r is *safe* if the following conditions hold: *(i)* for each global variable X of r there is a positive standard atom ℓ in the body of r such that X appears in ℓ, and *(ii)* each local variable of r appearing in a symbolic set $\{Terms{:}Conj\}$ also appears in a positive atom in $Conj$. A *weak constraint* (Buccafurri *et al*. 2000) ω is of the form:

$$:\sim b_1, \ldots, b_k, not\ b_{k+1}, \ldots, not\ b_m.\ [w@l]$$

where w and l are the weight and level of ω, respectively. (Intuitively, $[w@l]$ is read as "weight w at level l", where the weight is the "cost" of violating the condition in the body of w, whereas levels can be specified for defining a priority among preference criteria). An ASP program with weak constraints is $\Pi = \langle P, W \rangle$, where P is a program and W is a set of weak constraints. A standard atom, a literal, a rule, a program or a weak constraint is *ground* if no variables appear in it.

2.2 Semantics

Let P be an ASP program. The *Herbrand universe* U_P and the *Herbrand base* B_P of P are defined as usual. The ground instantiation G_P of P is the set of all the ground instances of rules of P that can be obtained by substituting variables with constants from

https://doi.org/10.1017/S1471068424000309 Published online by Cambridge University Press

U_P. An *interpretation* I for P is a subset I of B_P. A ground literal ℓ (resp., *not* ℓ) is true w.r.t. I if $\ell \in I$ (resp., $\ell \notin I$), and false (resp., true) otherwise. An aggregate atom is true w.r.t. I if the evaluation of its aggregate function (i.e., the result of the application of f on S) w.r.t. I satisfies the guard; otherwise, it is false. A ground rule r is *satisfied* by I if at least one atom in the head is true w.r.t. I whenever all conjuncts of the body of r are true w.r.t. I. A model is an interpretation that satisfies all rules of a program. Given a ground program G_P and an interpretation I, the *reduct* (Faber *et al.* 2011) of G_P w.r.t. I is the subset G_P^I of G_P obtained by deleting from G_P the rules in which a body literal is false w.r.t. I. An interpretation I for P is an *answer set* (or stable model) for P if I is a minimal model (under subset inclusion) of G_P^I (i.e., I is a minimal model for G_P^I) (Faber *et al.* 2011). Given a program with weak constraints $\Pi = \langle P, W \rangle$, the semantics of Π extends from the basic case defined above. Thus, let $G_\Pi = \langle G_P, G_W \rangle$ be the instantiation of Π; a constraint $\omega \in G_W$ is violated by an interpretation I if all the literals in ω are true w.r.t. I. An *optimum answer set* for Π is an answer set of G_P that minimises the sum of the weights of the violated weak constraints in G_W in a prioritised way.

2.3 Syntactic shortcuts

In the following, we also use *choice rules* of the form $\{p\}$, where p is an atom. Choice rules can be viewed as a syntactic shortcut for the rule $p \mid p'$, where p' is a fresh new atom not appearing elsewhere in the program, meaning that the atom p can be chosen as true.

3 Case study and proposed solution framework

In this section, we present our case study about dynamic traffic distribution for urban networks, and the solution framework we propose, in two separate subsections.

3.1 Problem description

Urban traffic routing aims to mitigate traffic congestion by navigating the vehicles and recommending less-congested routes, hence supporting a better use of the capacity of the urban network. Recent advances in connected autonomous vehicles (CAVs) technology provide an opportunity for routing approaches to be increasingly practicable and to revolutionise the field, as the communication capabilities of CAVs can allow roadside agents to collect real-time traffic information, and can support real-time communication between the vehicle and a centralised traffic control system (Shladover 2018; Vallati and Chrpa 2018). A centralised approach aims to provide the optimal routes for each CAV from the perspective of the traffic management centre, with the clear benefit of having a holistic vision of the controlled urban region. In other words, the centralised approach allows considering both the dynamic system optimal principle (Merchant and Nemhauser 1978) and the dynamic user optimal principle (Friesz *et al.* 1989) when routing vehicles, and the trade-off between the two according to traffic conditions.

In this work, we consider a centralised scenario where a controller oversees the network in real time. The controller has knowledge of the network structure, that can possibly

https://doi.org/10.1017/S1471068424000309 Published online by Cambridge University Press

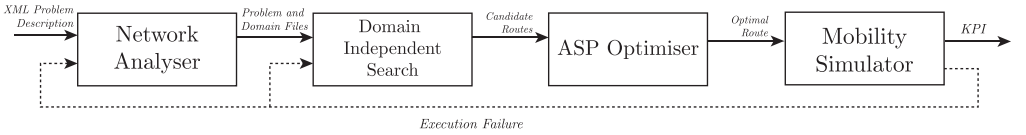


Fig. 1. The solution framework.

be updated according to accidents or other unexpected events. While in operation, the controller is provided with the list of incoming vehicles (controlled), and the position of the vehicles already navigating the network (hereinafter referred to as simulated). On the basis of this information, the controller provides an optimised route for controlled vehicles, that aims at minimising overall congestion while considering the dynamic aspects of traffic.

We assume that vehicles entering the controlled region communicate to the controller their destination and their current path, via an existing Vehicular Ad-hoc Network (VANET) (Cucor 2021), and the controller assesses the network status in terms of expected or recorded congestion, and returns a route to the vehicle under the form of a sequence of links to follow (Shahi *et al*. 2020). The vehicle then follows the given path, and such path can be considered by the traffic controller for informing the routing of future incoming vehicles.

It is worth highlighting that for the sake of traffic distribution and route optimisation, traffic signals are not explicitly modelled. This is established practice in the transportation field (see, e.g., (Bliemer and Raadsen 2020; Batista *et al.* 2021)) for a number of reasons. First, traffic signals can be implicitly modelled by considering the average time needed to navigate through a link and the corresponding junction to leave it: this is a very efficient way to reduce the complexity of the task. Second, the main driving factors of congestion in urban areas are demand, the structure of the network, and the capacity of links – in a sense, traffic signals are only ensuring that shared resources of the network are accessed correctly. Further, the setting of traffic lights is often unknown, even for those working on fixed-time mode, while reactive traffic control approaches implement algorithms to react to perceived traffic conditions, so counter engineering their behaviour is not trivial and adds significant computational complexity to the traffic distribution task.

3.2 The solution framework

Figure 1 presents our solution framework and its four components. In the following, we describe the first two components, that is, the Network Analyser and the Domain-Independent Search, while the last two components will be presented in Sections 4 and 5, respectively. The implemented framework, tailored to be used with the considered simulation and scenarios, is available at: https://github.com/matteocarde/tplp-traffic

3.3 Network analyser

The purpose of the Network Analyser component is to simplify the topology of the network. It takes several inputs, including the network structure represented as a XML file, the list of incoming new vehicles, and the positions of vehicles that already have

https://doi.org/10.1017/S1471068424000309 Published online by Cambridge University Press

a route within the network. For the sake of this implementation of the framework, all input files are in the format used by the SUMO simulator. The Network Analyser then generates a simplified logical representation that describes the network, creating a model of the search space where viable solutions can be found in the next phase.

To achieve this, the Network Analyser employs a *preprocessor* that is specifically designed to build an internal model of the road network. The preprocessor performs various tasks to simplify the network, for instance, (*i*) joining small streets together, (*ii*) simplifying intersections in roundabouts, and (*iii*) removing streets from the network that cannot be used by vehicles, such as no-traffic zones or streets reserved for public transportation. Regarding (*i*) and (*ii*), it is important to highlight a technical detail implemented by the processor. The output of the Network Analyser serves as input for the Domain-Independent Search component and, subsequently, to the ASP Optimiser component. Since these components need to capture the complete flow of traffic, they must be time-dependent. To achieve accurate time-dependent analysis, it is crucial to discretise time into steps that are the right trade-off: being small enough to capture real traffic nuances but large enough to allow for efficient planning. The problem of finding the right discretisation step when planning in temporal scenarios is a well-known problem in planning (see, e.g., (Gigante *et al*. 2022; Cardellini *et al*. 2024b)). In our case, a large discretisation step could greatly impact on the quality of the produced plan. To exemplify this, let us suppose that a vehicle is about to navigate two streets, s_1 and s_2, with s_1 leading to s_2. The time to navigate s_1, at a speed of 45km/h is of $14s$, and the time for s_2 is of $12s$. In an ideal settings with constant speed and continuous time, it would take a vehicle $26s$ in total to cross the 2 streets. If instead we account for a discretisation step of $10s$, both s_1 and s_2 will take 2 discretisation steps to cross, that is, $20s$, with a total time of $40s$. It is evident that large discretisation steps can lead to significantly misleading navigation times being considered. Further, with large discretisation steps, the times needed to cross streets tend to be the same, and thus the ASP Optimiser component could opt for solutions where the vehicles take routes with the less number of streets possible, but which in reality would amount to greater running times. Having a smaller discretisation step, like of $1s$, could guarantee that the solution is more cogent to the reality, but this would greatly increase the complexity of the optimisation problem and negatively affect the total solving time.

Moreover, this discretisation step has an impact on the network's topology. For example, in most of the existing traffic simulation tools, a roundabout comprises multiple small streets that connect all incoming and outgoing streets at various intersections. However, representing each small street individually during the optimisation phase would not be accurate, as each of these small streets would need to be run in a time equal to the discretisation step, resulting in a large simulated running time not representative of the real running time. To address this, the preprocessor combines the small streets that connect the incoming and outgoing streets of the roundabout, creating one longer street for each enter/exit combination of the roundabout, which can more realistically model the flow of traffic within the discretised time steps. As a result of this grouping, a single street is represented multiple times in the network. Ensuring that the total capacity of the roundabout is respected becomes the responsibility of the optimiser, which will ensure that the maximum number of vehicles inside the roundabout will be respected. Moreover, having

https://doi.org/10.1017/S1471068424000309 Published online by Cambridge University Press

removed any streets from the network that cannot be used by vehicles, such as no-traffic zones or streets reserved for public transportation, we might be left with intersections with only two intersecting streets and which can thus be joined to better deal with the discretisation step. In our experimental settings, after having performed the aforementioned simplifications, we performed a simulation analysis on how different discretisation steps would have affected the running times of a vehicle that needed to run through several runs and choose the one that allowed for a total running time as close as possible to the real running time. We thus chose the discretisation step of $5s$.

3.4 Domain-Independent Search

The Domain-Independent Search component receives the simplified network representation generated by the Network Analyser as input. Its main purpose is to identify and output suitable routes for vehicles entering the network and determine the time ranges in which these vehicles will enter or exit each street along their routes.

The origin and destination of each approaching vehicle are known in advance, and the framework aims to find high-quality routes that connect these two points. To achieve this, the search phase computes all possible (acyclic) paths in the network graph that connect the source and target streets for each entering vehicle. However, in large and complex maps, this would result in an unmanageable number of routes. To tackle this, for each vehicle approaching the network, we firstly employ a Dijkstra search algorithm to explore the network graph. Starting from the source street, for each adjacent street, we define a new route, composed of the source street and the adjacent street, labelled with the sum of their length. We insert all the generated routes into a priority queue, we select the route with the smallest length and, by concatenating each adjacent street of the final street of the route with the route itself, we keep iterating the procedure, exploring the network. When the target street is reached, due to the properties of the Dijkstra algorithm, we know to have found the shortest route between the source and the target street. However, differently from the standard Dijkstra approach, we don't conclude the search when the target street is found, but we simply save the route, not putting it back in the priority queue and keep exploring the network, saving a new route every time the target street is reached. A loop detection mechanism is put in place to detect when the concatenation of a new street would cause a cycle and simply discard the generated route. We stop the procedure when a desired number of (acyclic) routes is reached (in our experimental setting, 60 routes, appropriate for the sizes of the networks considered). These routes are guaranteed, by the Dijkstra algorithm, to be the shortest among all the possible routes but are not guaranteed to be the fastest, since congestion could change the running times of a street.

Unfortunately, these routes, computed for each approaching vehicle, could still be too many to be considered by the solver. Moreover, most of the routes could only differ slightly by a few streets and thus quite interchangeable in distributing the traffic. Instead, we are interested in having a smaller number of routes but "most different" between each other, that is, sharing the least number of streets possible. This way, the ASP solver could decide to direct a vehicle to a longer route, but which is less congested and actually faster to run through. To compute these "most different" routes for each pair of routes r_1 and

https://doi.org/10.1017/S1471068424000309 Published online by Cambridge University Press

r_2 we compute a similarity score through a function $\sigma(r_1, r_2) \in [0, 1]$. In our experimental setting, the function σ is computed as

$$\sigma(r_1, r_2) = \frac{|\text{streets}(r_1) \cap \text{streets}(r_2)|}{\min(|\text{streets}(r_1)|, |\text{streets}(r_2)|)},$$

where streets(r) is the set of the streets of a route r. Then, we create a set $\mathcal{R} = \{R_1, \ldots, R_q\}$ where each $R_i \in \mathcal{R}$ is a set of routes such that for each pair of routes r_1, r_2 we have $r_1, r_2 \in R_i$ if $\sigma(r_1, r_2) < \sigma_T$ and $r_1 \in R_i, r_2 \in R_j$ with $i \neq j$ if $\sigma(r_1, r_2) \geq \sigma_T$, with $\sigma_T \in [0, 1]$ being a similarity threshold (in our experimental setting, $\sigma_T = 0.5$). The size q of the set $\mathcal{R}$ depends on the instance but, usually, it is not greater than 3, since for every source street being at the edge of the network, there are usually three cardinal directions in which the vehicles can go, and thus three possible sets of "most different" routes. From each of the set R_i, we then select the top k routes inside R_i, ordered by length (in our experimental setting, $k = 5$, thus usually having 15 final routes to choose for each approaching vehicle). It is worth noting that the ASP Optimiser can therefore find the optimal route, according to the destination of the vehicle and the network condition, out of the identified k final routes. In principle, the optimal route selected by the ASP Optimiser can be different from the global optimal route: this is not deemed to be an issue in urban areas, where the highly dynamic nature of traffic plays a major role.

We will see in the encoding of the ASP Optimiser in the following section that, to deal with the temporal dimension, we will discretise time by a quantum (in our setting $5s$) and then emulate the evolution of the movement of each vehicle running its route, where vehicles can enter or exit a street only on times multiple of this quantum. For the rapid execution of the ASP Optimiser component, we compute in advance time ranges for when vehicles could enter and exit the streets along their routes. This computation serves two purposes: speeding up the simulation of traffic flow and reducing overall computation time. By utilising the knowledge of all the vehicles' routes within the network (as determined when the vehicles are approaching the network), it becomes possible to compute time ranges for the entrance and exit timings in every street of a potential route run by an approaching vehicle by solving two relaxed simulations for each vehicle:

1. the first simulation is performed assuming that no other vehicle is present in the network, that is, there is no congestion, and thus the vehicle can run at its maximum speed (45 km/h),
2. the other simulation is performed assuming that all other vehicles in the map occupy all the streets of their routes at the same moment, thus obtaining, for each street, the maximum amount of congestion possible. For each street, we then compute the speed of the vehicles, proportional to the level of congestion, and from that speed we get the time to traverse the street.

These two relaxed simulations provide an estimate of the minimum and maximum entry and exit time for each vehicle. Computing the speed proportional to the congestion can be done in several ways, in our experimental setting we perform it this way. We firstly compute the capacity of the street, which is the number of possible vehicles which could occupy the street. This number is proportional to the length of the street and the number of lines of the street. In our experimental settings, we make the assumption that a car

https://doi.org/10.1017/S1471068424000309 Published online by Cambridge University Press

occupies $8m$, that is, the sum of the length of a standard car, usually $5m$, and the safety distance of $3m$ in the front. In a real scenario, the safety distance depends on the speed of the car, which is, however, what we want to compute. Nevertheless, in an urban scenario, where vehicles are not allowed high speed, the safety distance of $3m$ was validated by traffic operators. We compute the capacity of a street s as:

$$\text{capacity}(s) = \left\lceil \frac{\text{lines}(s) \times \text{length}(s)}{8m} \right\rceil . \tag{1}$$

We select the rounded up value to avoid small streets to have a capacity of 0. We can now estimate the speed of the cars in the street s. Let vehicles(s) be the number of vehicles in the street s, that is, its congestion. The speed of a street s is estimated as:

$$\text{speed}(s) = \begin{cases} 45km/h & \text{if } 0 \leq \text{vehicles}(s) < 0.4 \times \text{capacity}(s), \\ 30km/h & \text{if } 0.4 \times \text{capacity}(s) \leq \text{vehicles}(s) < 0.7 \times \text{capacity}(s), \\ 15km/h & \text{if } 0.7 \times \text{capacity}(s) \leq \text{vehicles}(s). \end{cases} \tag{2}$$

While this approach is only an estimation, it was validated, together with the thresholds (i.e., 0.4 and 0.7), by traffic operators and gave good results in the experimental setting.

The selected routes for each approaching vehicle together with the minimum and maximum entry and exit times are then passed to the third component of the architecture, the ASP Optimiser, dealing with optimisation of the routes.

4 ASP for the optimisation phase

In the section, we detail the third component of the solution framework, referred to as ASP Optimiser, which has the role to select the best route for each vehicle entering the network. Specifically, the input of the component is a set of ASP facts, referred to, in the following, as Data Model, and it uses an ASP encoding to compute the best route for every vehicle approaching the network.

4.1 Data model

The data model consists of the following atoms:

- `streetOnRoute(S,R,MIN,MAX)` represents the relationship between street `S` and route `R`. The variables `MIN` and `MAX`, computed by the Search component, define the expected time range for a vehicle starting at $t = 0$ to traverse route `R` and enter street `S`.
- `link(S1,S2)` indicates that there is a connection from street `S1` to street `S2`.
- `vehicle(V,T)` defines the presence of a vehicle `V` of type `T` on the map. `T` can be either `con` (*controlled* vehicle) or `sim` (*simulated* vehicle). We remind that a controlled vehicle is a new vehicle entering the region for which a route has to be identified, while a simulated vehicle already has a set route, and the system only needs to track its presence on the map.
- `origin(V,FROM)` designates street `FROM` as the starting point of vehicle `V` during the planning process. For controlled vehicles, the origin represents the street where the

https://doi.org/10.1017/S1471068424000309 Published online by Cambridge University Press

vehicle first enters the map. Similarly, `destination(V,TO)` specifies street `TO` as the final destination that will take the vehicle `V` outside the map.

- `possibleRouteOfVehicle(V,R)` indicates that vehicle `V` can follow route `R` to move from its origin to its destination. For simulated vehicles, there will be only one possible route available, determined when the vehicle entered the network. For controlled vehicles, this atom represents a subset of all possible routes from the origin to the destination (as discussed in the previous section).
- `time(T)` represents the time unit used for scheduling, ranging from 0 to the maximum horizon when all vehicles have left the network.
- `capacity(S,N)` denotes the capacity of street `S`, with `N` indicating the capacity value, as computed by Eq. (1).
- `trafficTravelTime(K,S,T)` represents the time required to traverse street `S` under different traffic conditions. The variable `K` can take values of `heavy`, `medium`, or `low`, indicating the traffic amount. These amounts are correlated to the speeds of 15 km/h, 30 km/h, or 45 km/h, respectively, as computed by Eq. (2). The time `T` is simply computed by dividing the length of the street with the relative speed.
- `maxTrafficTravelTime(S,T)` models the time it would take to clear the street in case of congestion when it reaches its maximum capacity.
- `trafficThreshold(K,S,MIN,MAX)` defines the range between `MIN` and `MAX` of vehicles in street `S` that categorise $\texttt{K} \in \{\texttt{heavy}, \texttt{medium}, \texttt{low}\}$ traffic, corresponding to the traffic levels mentioned in Eq. (2) (e.g., if `K` is `medium` then $\texttt{MIN} = 0.4 \times \text{capacity}(s)$ and $\texttt{MAX} = 0.7 \times \text{capacity}(s)$).
- `roundabout(R,C)` indicates the existence of a roundabout `R` with a maximum capacity of `C`, computed as the sum of all capacities of the streets of the roundabout.
- `streetInRoundabout(SS,R)` indicates that the simplified (result of original street's grouping) street `SS` is part of the roundabout `R`.

The output of the ASP Optimiser consists of the following atoms:

- `solutionRoute(V,R)` assigns a route `R` to the vehicle `V`.
- `enter(V,S,IN)` and `exit(V,S,OUT)` specify the times when a simulated vehicle `V` enters and exits street `S`. We remind that an upper and lower bound of these values is determined in the Search phase and provided as input through the atom `streetOnRoute(.,.,.,.)`.

4.2 ASP encoding

Figure 2 presents the rules utilised in the ASP encoding. Rule r_1 defines the atom `solutionRoute`, which, for every *controlled* vehicle, selects exactly one route among the different selected routes (computed during the *Search* phase) that transport the vehicle from its origin to its destination. On the other hand, rule r_2 assigns the `solutionRoute` for *simulated* vehicles to be the route they are already following. Rule r_3 computes an entry time within the minimum and maximum range for each *controlled* vehicle, as determined in the *Search* phase. For the first street of origin, rule r_4 enforces an entry time of zero. Similarly, rule r_5 determines the exit time for vehicles at each street in their route. The atom `nVehicleOnStreet(S,T,N)` is defined by rule r_6 to count the

https://doi.org/10.1017/S1471068424000309 Published online by Cambridge University Press

```
{solutionRoute(V,R): possibleRouteOfVehicle(V,R)} = 1 :- vehicle(V,con).
solutionRoute(V,R) :- possibleRouteOfVehicle(V,R), vehicle(V,sim).
{enter(V,S,T) : time(T), T=MIN..MAX} = 1 :- vehicle(V,con), solutionRoute(V,R),
    streetOnRoute(S,R,MIN,MAX), not origin(V,S).
enter(V,S,0) :- origin(V,S).
{exit(V,S,T) : time(T), T=IN+1..IN+MAX} = 1 :- vehicle(V,con), enter(V,S,IN),
    maxTrafficTravelTime(S,MAX).
nVehicleOnStreet(S,T,N) :- enter(_,S,T), N = #sum{1,V: enter(V,S,IN), IN <= T; -1,V: exit(V,S,OUT),
    OUT <= T}.
travelTime(S,T,X) :- enter(_,S,T), nVehicleOnStreet(S,T,N), trafficThreshold(heavy,S,A,_), N >= A,
    trafficTravelTime(heavy,S,X).
travelTime(S,T,X) :- enter(_,S,T), nVehicleOnStreet(S,T,N), trafficThreshold(medium,S,A,B), N >= A, N
    < B, trafficTravelTime(medium,S,X).
travelTime(S,T,X) :- enter(_,S,T), nVehicleOnStreet(S,T,N), trafficThreshold(low,S,_,B), N < B,
    trafficTravelTime(low,S,X).
:- vehicle(V,con), exit(V,S,OUT), enter(V,S,IN), travelTime(S,IN,X), OUT < IN + X.
:- vehicle(V,con), exit(V,S1,OUT1), enter(V,S2,IN2), link(S1,S2), IN2 != OUT1.
:- enter(V,S,T), vehicle(V,con), capacity(S,MAX), nVehicleOnStreet(S,T,N), N > MAX.
:- enter(V,SR,T), streetInRoundabout(SR,R), vehicle(V,_), roundabout(R,MAX), #sum{X,S:
    nVehicleOnStreet(S,T,X), streetInRoundabout(S,R)} > MAX.
:~ nVehicleOnStreet(S,T,N). [N@2,S,T]
:~ destination(V,S), exit(V,S,T). [T@1]
```

Fig. 2. ASP encoding used during optimisation.

number of vehicles on street `S` at time `T`. This atom is then used in rules r_7 to r_9 to compute the `travelTime(S,T,X)` atom, representing the time (`X`) required for a vehicle to traverse the entire length of street `S` under different traffic conditions: heavy, medium, or low. These computations assume that the vehicle enters street `S` at time `T`. Constraint r_{10} enforces that once a vehicle enters a street, it must remain on it for at least the amount of time specified by the `travelTime(.,.,.)` atom. This constraint serves as a lower-bound restriction, as the vehicle may remain congested and depart later. Constraint r_{11} establishes an order among the streets within a route. Constraints r_{12} and r_{13} ensure that vehicles adhere to street capacities and roundabout restrictions, respectively. The weak constraint r_{14} optimises the number of vehicles on the street. Then, as the second optimisation criteria, weak constraint r_{15} imposes a preference for the solution that enables vehicles to reach their destinations more quickly. The ASP encoding and some solution examples can be found in the following repository: https://github.com/matteocarde/asp-traffic.

5 Experiments, monitoring, and execution

For the last phase of the framework (*Mobility Simulator*), we have employed SUMO (López *et al*. 2018), which is a state-of-the-art Urban Mobility Simulator to monitor the state of the network, execute the computed optimal solution plan, and verify its quality through the computation of Key Performance Indicators (KPIs), such as *Total Duration* (i.e., the time the last vehicle exits the network), *Average Route Length, Speed, Duration, Waiting Time* (i.e., the time vehicles spend in queues) and *Depart Delay* (i.e., the time vehicles spend waiting for the road to free to enter the network). This phase takes the responsibility for executing the optimal plan found by the ASP Optimiser and to account for all the real-world (*microscopic*) nuances not modelled in the previous phases. For example, we leave to this component the job of simulating the flow of traffic light at intersections, cadenced by the switching phases of traffic lights. The overtakes

https://doi.org/10.1017/S1471068424000309 Published online by Cambridge University Press

among vehicles, the use of lanes, and the order in which vehicles are queued in traffic, are other aspects overlooked in the previous phases, in which the streets are considered as buckets with no particular order between them and the managed aspect is how much each vehicle will occupy (on average) the single street.

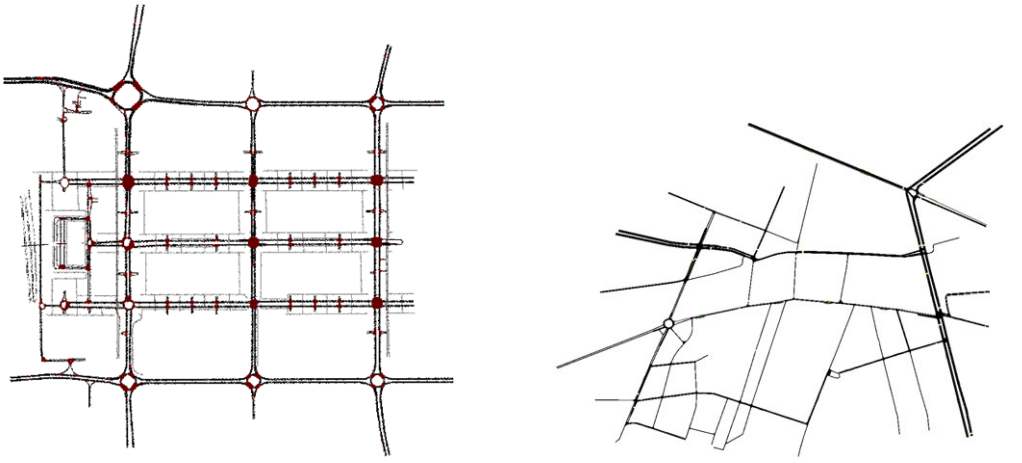
Given the fact that the previous steps reason with an abstracted representation of the problem to solve, it is possible that the execution of the plan leads to a state of the world that is significantly different from the expected one. For instance, some roads expected to have free flows of vehicles can be very congested, and there can be vehicles queuing at junctions because of traffic not leaving the controlled region at the expected rate. These discrepancies can be provided as feedback to the Domain-Independent Search, that can take them into account to define the current state of the network. Further, there may be disruptions that are modifying the viability of part of the network. For instance, a car accident can happen (they can be simulated in SUMO by modifying the behaviour of drivers), unexpected road works may be required, or extreme weather conditions can affect the area: this kind of events can reduce the capacity of roads, or completely block portions of the network, according to severity. This information about unexpected or unplanned events can be fed back to the Network Analyser to update its internal representation of the network and of the links, hence ensuring that the subsequent modules are reasoning upon a more accurate representation of reality. In this implementation we did not explicitly take into account these classes of events, as they are complex to simulate with SUMO and may require domain expertise to be properly modelled and encoded. However, the framework is designed to support this kind of updates, with no modification to the presented modules – the implementation of the feedback loop in a target area is left for future work.

5.1 Empirical settings and results

We assess the proposed approach using SUMO and by considering two scenarios: Bologna and Milton Keynes. The Bologna (Bieker *et al*. 2014) scenario considers the 7^{th} city in Italy by population (approx. 400, 000). The considered area was constructed around the "Andrea Costa" road in Bologna, in which the football stadium is located. The network, represented in Figure 3(b), includes more than 110 junctions and more than 170 links. The total length of the modelled links is more than 33 kilometres. The scenario includes the demand for Bologna's peak hour (8am – 9am) in which 8, 620 vehicles roam the network, in some days of November 2008. The interested reader is referred to Bieker *et al*. (2014) for a detailed description of how the network was constructed.

The Milton Keynes scenario considers the largest town in Buckinghamshire, United Kingdom, with a population of approximately 230, 000. A diagram of the considered network is shown in Figure 3(a): it covers an area of approximately 2.9 square kilometres. The network includes more than 25 junctions and more than 50 links. The total length of the modelled links is more than 45 kilometres. The model simulates the morning rush hour, and has been built by considering historical traffic data collected between 8am and 9am on non-holiday weekdays. Data has been provided by the Milton Keynes Council, and gathered by sensors distributed in the region between December 2015 and December 2016. Traffic signal control information has been provided by the Council. The model has

https://doi.org/10.1017/S1471068424000309 Published online by Cambridge University Press



Milton Keynes Bologna

Fig. 3. The considered SUMO model of the central Milton Keynes (left) and Bologna (right) urban areas. Please note that the maps are not in scale, so can not be directly compared.

been calibrated and validated. During the morning rush hour, 1,900 vehicles are entering the controlled region, and the main traffic flows are from North to South-East, and from West to East. This is because large residential areas are located at the North and West of the modelled region.

Our approach uses the TraCI interface[1] (Wegener *et al.* 2008) to interact with the SUMO simulation environment, to get the current network status, communicate with approaching vehicles, and inform vehicles of re-routing. Every time a new vehicle approaches the network, our approach is called, and a route is provided to the new entering vehicle. In both scenarios, the simulation is run until all the vehicles left the network. For each set of experiments, the simulation is run five times and results are averaged. All the experiments were run on a MacBook Pro with a 2.5 GHz Intel Core i7 quad-core, with 16 GB of RAM.

To empirically assess the performance of the proposed framework, in this subsection we compare actual simulated traffic data of the Milton Keynes and Bologna urban areas with a simulation in which the same vehicles are routed using our proposed approach, where k has been set to 5 as previously discussed. Table 1 shows a comparison between the two approaches in terms of a number of traditionally considered KPIs, that focus on delay, waiting time, speed, and path duration. As it can be seen from the comparison, in the Milton Keynes urban area, the proposed approach can greatly increase the overall performance of the network, spreading traffic and reducing congestion, increasing the average speed of vehicles and allowing the network to free faster. In Bologna, instead, the KPIs are only slightly increased with respect to the ones computed on actual traffic data, but still showing how our proposed approach can capture all the nuances of urban traffic control and to deal with the risk of congesting the network. The high differences in improvement which can be seen in Milton Keys with respect to Bologna can be explained

[1] https://sumo.dlr.de/docs/TraCI.html

https://doi.org/10.1017/S1471068424000309 Published online by Cambridge University Press

Table 1. *Performance of actual traffic data coming from the Milton Keynes and Bologna's urban area, and the same vehicles routed using our proposed approach*

	Milton Keynes		Bologna	
	Actual	Our approach	Actual	Our approach
Total duration [s]	15729	5065	5692	5647
Avg. route length [m]	2465	2107	1636	1678
Avg. speed [m/s]	2.49	5.28	6.37	6.58
Avg. duration [s]	3718	515	283	281
Avg. waiting time [s]	3132	259	97	95
Avg. depart delay [s]	791	55	192	188

by the two very different topologies of the networks. As it can be seen, in Milton Keynes (Figure 3(a)) the streets form a sort of *Manhattan Grid* in which parallel streets are more or less equal in terms of number of lanes, length, and intersections. For this reason, a car entering the network has a plethora of possible routes to choose, which are more or less of the same length, giving the possibility to better spread the traffic through the whole map. In Bologna (Figure 3(b)), instead, it can be noted how streets in the outer ring are more structured to deal with high volumes of traffic (with a large number of lanes and roundabouts to reduce traffic), while streets near the centre of the map (which constitute the residential area) have mostly one lane and many intersections. For this reason, vehicles have a smaller number of promising routes to choose from, leaving no choice to the planner but to let vehicles move through the streets of the outer ring.

5.2 ASP evaluation

We now evaluate the ASP encoding presented in Figure 2 in terms of capacity to rapidly find a high-quality solution to the routing of new vehicles inside the network. Every time one vehicle approaches the network, the Network Analyser calls the Domain-Independent Search, and then the ASP Optimiser is invoked to find optimal routes for the approaching vehicles. The discretisation step has been set to 5 s according to some preliminary experiments. As ASP Optimiser, we employed CLINGO configured with the option `--parallel-mode = 2` which executes the task in two threads, one using the Branch and Bound algorithm (Gebser *et al*. 2015), and one using the Unsatisfiable Core algorithm (Andres *et al*. 2012). This approach has already been proven effective in other application domains (Cardellini *et al*. 2024a). Figure 4 presents the scalability results and dimensionality for the network of Milton Keynes and Bologna. It shows the correlation between the solving time of CLINGO (Top) and the number of vehicles which roam the network (Bottom). The x-axis is grouped in bins of 50, and the histogram below shows the number of instances (i.e., each time a new vehicle enters the network) for each bin. The instances of Milton Keynes and Bologna with less than 600 vehicles can all be solved optimally within 30 s (actually, in the majority of cases, in less than 10 s). When the dimensionality increases, the solver can still find optimal solutions in most of the cases (the median of the boxplot is always under 30 s), but for some instances the cut-off time

https://doi.org/10.1017/S1471068424000309 Published online by Cambridge University Press

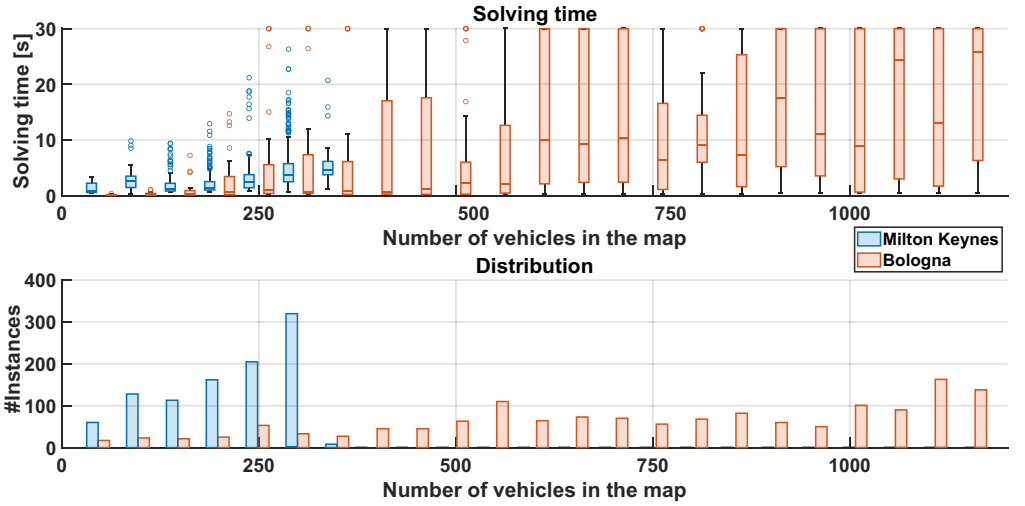


Fig. 4. (Top) boxplot of the solving time of CLINGO in correlation with the number of vehicles inside the networks. (Bottom) histogram representing the number of instances (i.e., each time a new vehicle enters the network) w.r.t. the number of vehicles inside the map. In the two charts, the x-axis has been clustered in bins of 50.

is reached, and a suboptimal solution is returned. As shown in Table 1, this approach can still improve the performances inside the networks.

6 Related Work

One of the most common methods in the literature to optimise traffic flow in road networks is to schedule traffic signal switching phases, or *signal phase plans*, with the aim of minimising vehicles' delay (Papageorgiou *et al*. 2003; Dotoli *et al*. 2006). The well-known real-time adaptive traffic control system SCOOT (Bretherton 1990), for example, makes use of this methodology and is used extensively throughout Europe and the United Kingdom. Unfortunately, managing only the switching phases of traffic lights has some limitations: the only metric which is controllable and optimisable is the waiting time at intersections (which has a direct impact on the total travel time of vehicles) but is not straightforwardly expandable to consider other metrics (e.g., fuel consumption). Moreover, the (*macroscopic*) point of view of traffic lights, which manages the flow of traffic modelling incoming and outgoing lanes as queues of vehicles, does not allow for a more detailed (*microscopic*) consideration of the single vehicles and their routes inside the network. Microscopic simulation models have been largely discarded in the literature due to their high complexity and low scalability. In this paper, we leverage ASP, coupled with domain-dependent optimisations, to model the traffic flow of hundreds of vehicles inside large European cities from a *microscopic* perspective.

The use of AI techniques in road transportation was shown to be effective in optimising traffic flows (Miles and Walker 2006; Abduljabbar *et al*. 2019). For instance, Vallati et al. (2016) introduced a *mixed discrete-continuous planning* (Fox and Long 2006) approach for reducing congestion through traffic signal optimisation, that was

https://doi.org/10.1017/S1471068424000309 Published online by Cambridge University Press

subsequently improved to be deployed in urban areas of the United Kingdom (McCluskey and Vallati 2017; El Kouaiti *et al*. 2024). Chrpa *et al*. (2016), instead, used a *temporal planning* approach for managing traffic, through a *microscopic* perspective, intending to reduce air pollution and respect air quality limitations. Other instances of urban traffic problems solved with automated planning technologies can be found in Cenamor *et al*. (2014). Even if automated planning has been beneficial in efficiently solving several real-world problems in transportation (Cardellini *et al*. 2021; Ramírez *et al*. 2018), the main point of failure is the ability to scale in the presence of large number of vehicles and the limited capabilities of generating optimised solutions, which are instead not issues for our approach.

Noteworthy, ASP has already been used in this context. Eiter *et al*. (2020) improve SCOOT for dynamically adjusting traffic signals via ASP. They employ SUMO for simulation, but do not rely on historical data. Our works focus on different aspects of traffic control, that is, the distribution of connected vehicles. Cardellini *et al*. (2023) proposed a framework to reason upon risks and their mitigation in the distribution of vehicles in urban areas. In railway, Abels *et al*. (2021) deal with the train scheduling problem, with a hybrid solution as done by Ramírez *et al*. (2018), which integrates ASP and difference logic, tested on real-world instances crafted by domain experts at Swiss Federal Railways. Beck *et al*. (2012) use ASP to address the management of inconsistent traffic regulations, such as incorrect sign postings or software errors during data acquisition.

7 Conclusion

In this paper, we presented a framework for dealing with the problem of dynamic traffic distribution for urban networks. The framework is composed of four phases: Network Analyser, Domain-Independent Search, ASP Optimiser and Mobility Simulator. The third phase of the approach relies on ASP for the computation of the optimal routes. The whole framework has been assessed on real-world data from two urban areas of the UK (Milton Keynes) and Italy (Bologna). Results show that the framework can significantly improve the considered traffic KPIs, and that the advantages depend also on the topology of the networks. Further, the contribution ASP can give is evaluated in terms of performance and metrics: results outline that all instances of Milton Keynes and Bologna up to 600 vehicles inside the network are solved, optimally, in a short time.

We see several avenues for future work. First, we are interested in assessing the framework in different urban areas, and in enhancing the feedback capabilities by considering a number of critical failures and their impact. Second, we plan to extend the proposed approach to deal with vehicles that do not follow the provided instructions. Finally, we are interested in integrating the proposed framework into the larger urban traffic control infrastructure, so that information about calculated traffic flows can be exploited to inform traffic authorities policies and actions.

References

Abduljabbar, R., Dia, H., Liyanage, S. and Bagloee, S. A. 2019. Applications of artificial intelligence in transport: An overview. *Sustainability* 11, 1, 189.

https://doi.org/10.1017/S1471068424000309 Published online by Cambridge University Press

Abels, D., Jordi, J., Ostrowski, M., Schaub, T., Toletti, A. and Wanko, P. 2021. Train scheduling with hybrid answer set programming. *Theory and Practice of Logic Programming* 21, 3, 317–347.

Andres, B., Kaufmann, B., Matheis, O. and Schaub, T. 2012. Unsatisfiability-based optimization in clasp. In *Proc. of ICLP Technical Communications*, volume 17 of *LIPIcs*, 211–221, Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik

Baral, C. 2003. *Knowledge Representation, Reasoning and Declarative Problem Solving.* Cambridge University Press.

Batista, S., Leclercq, L. and Menendez, M. 2021. Dynamic traffic assignment for regional networks with traffic-dependent trip lengths and regional paths. *Transportation Research Part C: Emerging Technologies* 127, 103076.

Beck, H., Eiter, T. and Krennwallner, T. 2012. Inconsistency management for traffic regulations: Formalization and complexity results. In *Proc. of JELIA*, volume 7519 of LNCS, Springer, 7519, 80–93, LNCS

Bieker, L., Krajzewicz, D., Morra, A. P., Michelacci, C. and Cartolano, F. 2014. Traffic simulation for all: a real world traffic scenario from the city of Bologna. *Modeling Mobility with Open Data: 2nd SUMO Conference 2014*.

Bliemer, M. C. and Raadsen, M. P. 2020. Static traffic assignment with residual queues and spillback. *Transportation Research Part B: Methodological* 132, 303–319.

Bretherton, R. 1990. SCOOT urban traffic control system – philosophy and evaluation. *IFAC Proceedings Volumes* 23, 2, 237–239.

Brewka, G., Eiter, T. and Truszczynski, M. 2011. Answer set programming at a glance. *Communications of the ACM* 54, 12, 92–103.

Buccafurri, F., Leone, N. and Rullo, P. 2000. Enhancing Disjunctive Datalog by Constraints. *IEEE Transactions on Knowledge and Data Engineering* 12, 5, 845–860.

Calimeri, F., Faber, W., Gebser, M., Ianni, G., Kaminski, R., Krennwallner, T., Leone, N., Maratea, M., Ricca, F. and Schaub, T. 2020. ASP-Core-2 input language format. *Theory and Practice of Logic Programming* 20, 2, 294–309.

Cardellini, M., De Nardi, P., Dodaro, C., Galata, G., Giardinii, A., Maratea, M. and Porro, I. 2024a. Solving rehabilitation scheduling problems via a two-phase asp approach. *Theory and Practice of Logic Programming* 24a, 344–367.

Cardellini, M., Dodaro, C., Maratea, M. and Vallati, M. 2023. A framework for risk-aware routing of connected vehicles via artificial intelligence. In *2023 IEEE 26th International Conference on Intelligent Transportation Systems (ITSC)*, IEEE, 5008–5013.

Cardellini, M., Maratea, M., Percassi, F., Scala, E. and Vallati, M. 2024b.Taming discretised PDDL+ through multiple discretisations. *Proceedings of the Thirty-Fourth International Conference on Automated Planning and Scheduling, ICAPS*, 59–67.

Cardellini, M., Maratea, M., Vallati, M., Bpleto, G. and Oneto, L. 2021. In-station train dispatching: A PDDL+ planning approach. In *Proc. of ICAPS.*, AAAI Press, 450–458.

Cenamor, I., Chrpa, L., Jimoh, F., Mccluskey, T. L. and Vallati, M. 2014. Planning & scheduling applications in urban traffic management. In *Proc. of PlanSIG*.

Chrpa, L., Magazzeni, D., Mccabe, K., Mccluskey, T. L. and Vallati, M. 2016. Automated planning for urban traffic control: Strategic vehicle routing to respect air quality limitations. *Intelligenza Artificiale* 10, 2, 113–128.

Cucor, B. 2021. Outlines of vehicular ad-hoc networks. *Proc. of TRANSCOM, Transportation Research Procedia* 55, 1312–1319.

Dotoli, M., Fanti, M. P. and Meloni, C. 2006. A signal timing plan formulation for urban traffic control. *Control Engineering Practice* 14, 11, 1297–1311.

Eiter, T., Falkner, A. A., Schneider, P. and Schüller, P. 2020. ASP-Based Signal Plan Adjustments for Traffic Flow Optimization. In *Proc. of ECAI*, volume 325 of *FAIA* IOS Press, 325, 3026–3033

El Kouaiti, A., Percassi, F., Saetti, A., Mccluskey, T. L. and Vallati, M. 2024. PDDL+ models for deployable yet effective traffic signal optimisation. In *Proceedings of the International Conference on Automated Planning and Scheduling*, 34, 168–177.

Faber, W., Pfeifer, G. and Leone, N. 2011. Semantics and complexity of recursive aggregates in answer set programming. *Artificial Intelligence* 175, 1, 278–298.

Fox, M. and Long, D. 2006. Modelling mixed discrete-continuous domains for planning. *Journal of Artificial Intelligence Research* 27, 235–297.

Friesz, T. L., Luque, J., Tobin, R. L. and Wie, B.-W. 1989. Dynamic network traffic assignment considered as a continuous time optimal control problem. *Operations Research* 37, 6, 893–901.

Gebser, M., Kaminski, R., Kaufmann, B., Romero, J. and Schaub, T. 2015. Progress in clasp Series 3. In *Proc. of LPNMR*, volume 9345 of LNCS, Springer, 368–383

Gelfond, M. and Lifschitz, V. 1991. Classical negation in logic programs and disjunctive databases. *New Generation Comput* 3, 4, 365–386.

Gigante, N., Micheli, A., Montanari, A. and Scala, E. 2022. Decidability and complexity of action-based temporal planning over dense time. *Artificial Intelligence* 307, 103686.

Lopez, P.Á., Behrisch, M., Bieker-Walz, L., Erdmann, J., Flotterod, Y., Hilbrich, R., Lucken, L., Rummel, J., Wager, P. and Wiebner, E. 2018. Microscopic traffic simulation using SUMO. In *Proc. of ITSC*, IEEE, 2575–2582.

Max Roser, H. R. and Ortiz-Ospina, E. 2013. World population growth.

Mccluskey, T. and Vallati, M. 2017. Embedding automated planning within urban traffic management operations. In *Proc. of ICAPS*, 391–399.

Merchant, D. K. and Nemhauser, G. L. 1978. A model and an algorithm for the dynamic traffic assignment problems. *Transportation Science* 12, 3, 183–199.

Miles, J. and Walker, A. J. 2006. The potential application of artificial intelligence in transport. In *IEE Proceedings-Intelligent Transport Systems*, 153, 183–198, Iet.

Niemela, I. 1999. Logic programs with stable model semantics as a constraint programming paradigm. *Annals of Mathematics and Artificial Intelligence* 25, 3–4,241–273.

Papageorgiou, M., Diakaki, C., Dinopoulou, V., Kotsialos, A. and Wang, Y. 2003. Review of road traffic control strategies. *Proceedings of the IEEE* 91, 12, 2043–2067.

Ramirez, M., Papasimeon, M., Lipovetzky, N., Benke, L., Miller, T., Pearce, A. R., Scala, E. and Zamani, M. 2018. Integrated hybrid planning and programmed control for real time UAV maneuvering. In *Proc. of AAMAS*, ACM, Richland, SC, USA, 1318–1326.

Shahi, G. S., Batth, R. S. and Egerton, S. 2020. MRGM: An adaptive mechanism for congestion control in smart vehicular network. *International Journal of Communication Networks and Information Security* 12, 2, 273–280.

Shladover, S. E. 2018. Connected and automated vehicle systems: Introduction and overview. *Journal of Intelligent Transportation Systems* 22, 3, 190–200.

Vallati, M. and Chrpa, L. 2018. A principled analysis of the interrelation between vehicular communication and reasoning capabilities of autonomous vehicles. In *Proc. of ITSC*, 3761–3766.

Vallati, M., Magazzeni, D., Schutter, B. D., Chrpa, L. and Mccluskey, T. L. 2016. Efficient macroscopic urban traffic models for reducing congestion: A PDDL+ planning approach. In *Proc. of AAAI*, AAAI Press, 3188–3194.

Van Mierlo, J., Maggetto, G., Burgwal, E. and Gense, R. 2004. Driving style and traffic measures - influence on vehicle emissions and fuel consumption. *Institution of Mechanical Engineers Part D Journal of Automobile Engineering* 218, 1, 43–50.

Wegener, A., Piórkowski, M., Raya, M., Hellbrück, H., Fischer, S. and Haubaux, J.-P. 2008. TraCI: an interface for coupling road traffic and network simulators. In *Proc. of the Communications and Networking Simulation Symposium*, 155–163.

https://doi.org/10.1017/S1471068424000309 Published online by Cambridge University Press