Urban Traffic Control Assisted by AI Planning and Relational Learning

Alberto Pozanco and **Susana Fernández** and **Daniel Borrajo**
Departamento de Informática, Universidad Carlos III de Madrid
Avda. de la Universidad, 30. 28911 Leganes (Madrid). Spain
apozanco@pa.uc3m.es, sfarregu@inf.uc3m.es, dborrajo@ia.uc3m.es

Abstract

Urban Traffic Control is a key problem for most big cities. An inefficient traffic control system can lead to increased traffic congestions that degrade city quality metrics such as average travel time or city pollution. Most common approaches focus on controlling traffic by appropriately setting traffic lights. Current systems in operation range from static control of traffic light phases to adaptive systems based on numeric models. In this paper, we propose an autonomic approach based on declarative automated planning to generate control plans only when the default behavior should be overridden. Planning is complemented with plan execution control and monitoring, replanning, as well as self-adaptive behavior using Relational Learning. Learning is used to anticipate the appearance of congestions and correctly solve them. Our system outperforms static approaches as well as a planning-based system that recently won a competition on autonomic behavior in Urban Traffic Control.

1 Introduction

Traffic efficient management and control in urban networks is an important challenge for city authorities. They usually want to achieve a variety of policy-based objectives, such as reducing atmospheric pollution or mitigating the effects of unexpected situations like accidents or road closure. There are many ways to set the traffic lights programs, ranging from early static off-line approaches, to most recent adaptive approaches that change the programs according to the state of the city. The reader is directed to surveys in the area [Papageorgiou *et al.*, 2007; Hamilton *et al.*, 2013].

From a centralized perspective, Automated Planning (AP) has been recently shown to perform well in this kind of tasks [Gulič *et al.*, 2015; Vallati *et al.*, 2016]. The main advantage of using AP is that the domain and problem descriptions are specified in a declarative language. Thus, even traffic engineers can easily include new actions, sensor information or metrics. Also, these models can be automatically updated by using learning techniques. In this paper we propose an approach that integrates a planning system for controlling traffic lights with a learning system that predicts when a street density is going to be high in the near future. In those cases, our system anticipates future problems by generating new goals to the planning module and starts a planning-execution-monitoring process. The proposed system can be seen as an instance of a full autonomic (autonomous) system, given that it incorporates many self-* properties, as self-monitoring (continuous observation), self-diagnosis (detects undesired behavior), self-optimization (planning), self-healing (executes actions) and self-adaptation (learning).

The paper is organized as follows: the next section describes the system architecture that integrates learning with AP; the third section formally defines AP tasks and describes the traffic-control domain; the fourth section briefly describes the learning system; the fifth section presents the experimental results; and the last section draws conclusions and outlines future work.

2 Architecture

We propose to use a planning-execution-monitoring architecture called PELEA to provide a framework that can integrate the various components of our system [Guzmán *et al.*, 2012]. Figure 1 shows a sketch of the architecture. At start, the *Execution* module receives an AP domain and problem. Then, it captures the current state of the world, *state*, and sets the problem initial state. The initial goal set could be also set by the *Goal&Metrics Generation* module. The *Monitoring* module calls the *Planning* module to obtain a plan whose actions are sent back to the *Execution* module. Once the actions are executed, the *Monitoring* module receives the necessary knowledge (current state, problem and domain) from the *Execution* module to initialize a new planning-execution-monitoring cycle. If the execution did not produce the expected changes (reduction in traffic density in some streets), it will result in the generation of new goals and a new initial state for a new call to the planner. The *Goal&Metrics Generation* module combines these goals with possible external ones (as the ones given directly by traffic controllers) to update the problem. The environment can be substituted by a *Simulator* in some domains, as the one we focus in this paper.

One of the greatest challenges in the proposed architecture is the generation of new goals. Here, we propose to apply machine learning techniques to infer when new goals should be generated to anticipate future problematic streets. In a training step, examples are generated by observing the traffic be-

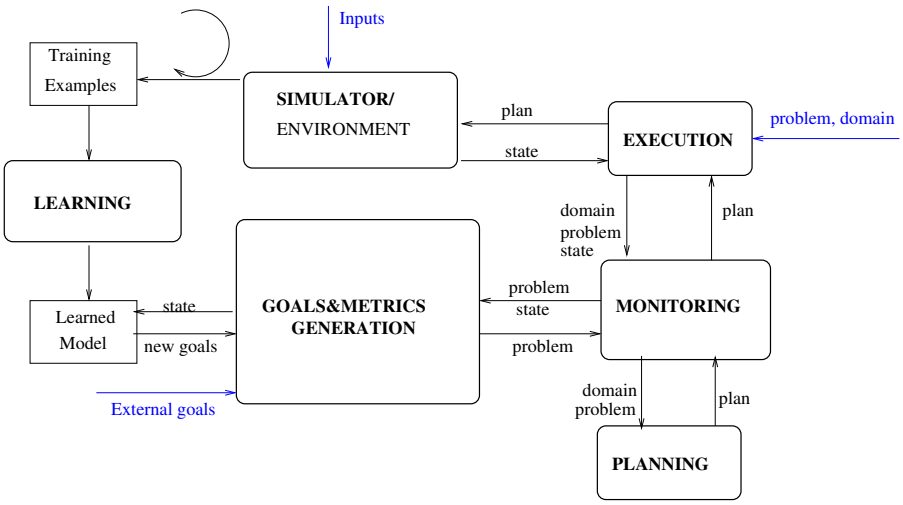

Figure 1: Planning and execution architecture that includes learning capabilities.

havior during some time periods, under different traffic conditions. Then, a learning algorithm can generate a model from those examples, such that given any new state it returns new goals. We are assuming here that the learning process is performed off-line, prior to the actual use of the AP-based system, but it could also be done on-line. The following section formally defines AP tasks and describes the Urban Traffic Control (UTC) domain we are using on this work.

3 Planning Tasks

In order to represent planning tasks compactly, the AP community uses the standard language PDDL (Planning Domain Description Language) [Fox and Long, 2003]. Most planners automatically generate an instantiated planning task from the PDDL declarative description of a domain D and a problem P. The domain defines the predicates for representing states and the actions that agents can perform. Figure 2 shows an example of an action in the domain definition. The problem describes the task to be solved at each reasoning step; i.e., the objects involved (e.g., streets, traffic lights), the initial state and the set of goals to achieve. Figure 3 shows a subset of a problem definition. The planner will receive both the domain and the problem files as input and it will try to find a solution plan for the given problem. In this case, the output of the planner will be a set of actions to be performed over the traffic lights, such that these actions override the default control program for a certain time period. If the planner has solved the congestion at the next reasoning step, the default program will take the control again. Otherwise, the next actions of the previously generated plan are executed.

This planning model assumes the world is deterministic and the agent has full observability, among other assumptions. In most real-world environments, this is not the case. Actions have stochastic outcomes (the traffic density is not always reduced in the same way when setting a longer green phase in a traffic light), and agents have partial observability (they do not know what the density due to new vehicles entering the city is going to be in the following time steps). There have mainly been two ways to handle uncertainty. In the first type of models, uncertainty is represented explicitly in the planning model and planners reason with those stochastic models [Bonet and Geffner, 2005]. In the second, planners reason with deterministic world models and when execution of some actions fails, the agent replans [Yoon *et al.*, 2007]. In this paper, we will use the second alternative given that, from a practical perspective, it is good enough for the domain we are focusing on.

4 Learning Traffic Behavior

In this section we define the task of learning when goals will arrive; that is, predicting the density level of the streets so we can anticipate their congestion, generating the appropiate goals for the planner. We formulate this problem as a time series prediction one, using Relational Learning in this case. Relational Learning is a Machine Learning technique that can capture the correlations between connected elements. In our case, we conjecture that the structured layout of a city can influence the density levels of some streets based on the ones that are connected to some others. Thus, it is a relational domain. Relational Learning also suits AP, because it allows induction over structured examples that can include first-order logical representations, like the ones used in PDDL.

4.1 Representation

The representation is based on a subset of the predicates we use in the planning traffic domain. In order to represent the time steps, we modify some of these predicates, adding the corresponding time steps. The predicates used for the learning task are shown in Table 1.

We distinguish two types of predicates: the static and the dynamic ones. The static part of the city is represented by the *connection* predicate, that indicates that a vehicle can move from one street section to another. All the *connection* predicates together represent the entire city network. The dynamic

```
(:action hm-green-to-all-ways
 :parameters (?t - traffic-light ?c - crossing ?sin - street
              ?sout1 - street ?sout2 - street ?sout3 - street)
 :precondition (and (goes-into ?sin ?c)
                    (goes-out ?sout1 ?c)
                    (traffic-lights-from-street ?t ?c ?sin)
                    (not (opposite-direction ?sin ?sout1))
                    (densityLevel ?sout1 moderate)...)
 :effect (and (not (state-to-street ?t ?sout1 red))
              (densityLevel ?sin low)...)
```

Figure 2: Part of an example description of a PDDL action.

```
(define (problem traffic1) (:domain traffic)
  (:objects s1 ... s566 - street
            c1 ... c30 - crossing
            tl1 ... tl10 - traffic-light)
  (:init (goes-into s1 c3)
         (opposite-directions s5 s7)
         (state-from-street tl1 s7 green)
         (densityLevel s1 high)...)
  (:goal (and (densityLevel s4 low)
              (densityLevel s35 low) ...)))
```

Figure 3: Part of an example PDDL problem file.

Predicate	Type
density(st,l)	Dynamic
connection(st,st)	Static
open$X$(tl,st)	Dynamic
densityLX(st)	Dynamic

Table 1: Predicates used in the learning task. X represents the time step. L represents the density level.

part of the city is formed by the state of the traffic lights and the density of the streets. The *openX(tl,st)* predicate represents a green traffic light *tl* located at street *st* at time step X. In our approach, X can take the values from one to three (X previous time steps, or time windows), but it is a parameter that can be modified to extend or reduce the prediction horizon. The *densityLX(st)* predicate indicates that a street *st* has a density level L at time step X. L can take the values *veryhigh, high, moderate, low* and *verylow*. The last predicate of each example, *density(st,l)*, represents the current density level *l* of the street *st*. This will represent the class of each example.

4.2 Algorithms

We are using TILDE [Blockeel and De Raedt, 1998] to learn relational decision trees. It receives two files as input: the settings file, where the user can specify the algorithm parameters, as well as defining the predicates and classes; and the knowledge base file, where both the training and test data are included. The output of the learning algorithm is a file containing the resulting relational tree and its translation into rules. It also contains the confusion matrix for the training and test sets. An example output of TILDE is shown in Figure 4, where A represents the example id and the other letters the predicates' arguments (B is the street whose density level, C, we want to predict). A minus symbol predating a variable means that it is new in the tree, while when the variable appears alone, it has to be referenced before. The classes to predict appear in the leaf nodes of the tree between brackets. For example, in the model shown in Figure 4, a high density would be predicted for a street B in two cases: (1) if its density was low two time steps ago, but there exists another street D connected to B whose density was high three time steps ago and was not low in the last time step; and (2) if its density was not low neither two time steps ago nor one time step ago.

```
density(-A,-B,-C)
densityLow2(A,B)?
+-yes:  densityHigh3(A,-D)?
        +-yes: connection(A,B,D)?
               +-yes: densityLow1(A,D)?
                      +-yes:[low]
                      +-no:[high]
               +-no:[low]
        +-no:[low]
+-no: densityLow1(A,B)?
      +-yes: [low]
      +-no: [high]
```

Figure 4: Example of TILDE output.

5 Experiments and results

On this work we use SUMO [Behrisch *et al.*, 2011], an open source traffic simulator developed by the German Aerospace Center (DLR). It allows to import or generate not only road networks, but also traffic demand. And it also allows users to define traffic lights control programs. We want to test first if we are able to build a model to predict the appearance of goals in advance, and then we try to apply the created model to several urban traffic control scenarios.

5.1 Results on Learning Goals

We are using a real city network in our learning experiments; a grid-like section of Houston downtown, shown in Figure 5. It is composed of 35 junctions, 140 traffic lights and 164 street sections. We have selected five particular street sections to learn from (A to E). We chose these city points due to their different traffic characteristics. C and D are street sections close to a Job Center. B is a point between the Job Center and the main exit of the city. E represents a street section far from the main traffic, while A is a random point with no specific features.

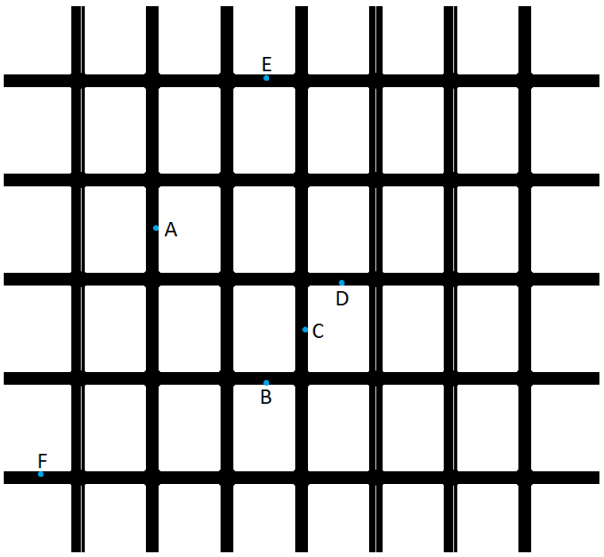

Figure 5: Benchmark network in SUMO. Models are created for points A, B, C, D and E. We assume that a Job Center is located on D. F corresponds to the main exit point of the city.

We have also defined a traffic demand that tries to emulate the real traffic flow of a city for an entire week. So, we define lower vehicles traffic at night, more traffic at rush hours, and higher traffic during week days than in the weekend. The Job Center is included, where most of the cars want to go during the work hours and also a main exit point, to go out of the city at the end of the workday. The rest of the routes are randomly generated. The vehicles may enter the city by any street section and can finish their trip in an inner (parking, mall, office...) or outer point of the network. A summary of the full traffic demand specification is shown in Figure 6.

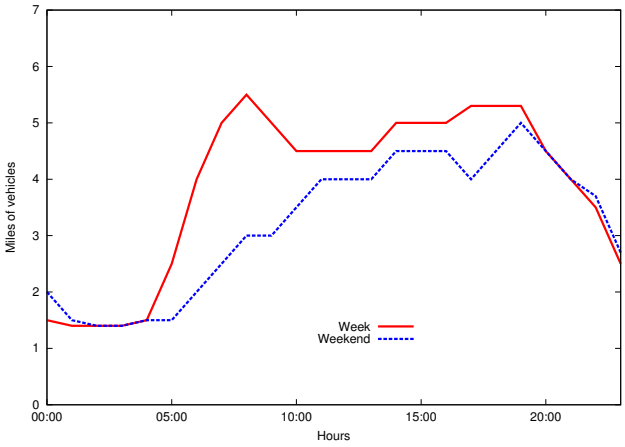

Figure 6: Summary of the generated traffic flows on weekdays and weekends. The y axis represents the number of vehicles that enter the network at each hour, in thousands, and the x axis represents the hours.

Data is collected every five minutes for the learning task, which means 2013 instances for the whole week. Five minutes is what we call "time step", the sample frequency. We have chosen this sample time as we want to collect traffic data from an entire week, and, at the same time, we want to keep a not very high number of instances so that TILDE is able to handle them. In our experimental setting, a step in the simulation corresponds to a second. Each instance stores the static part of the city previously described, as well as the dynamic component of the state in the last three time steps. We learn one relational model for each street section shown in Figure 5, and then we test with data of the other street sections.

We have also varied the density levels, both in the classes to predict and the predicates used on each instance. We have used two approaches. One is based on five density levels: *veryhigh, high, moderate, low* and *verylow*. A second version uses only two: *high* and *low*. All the generated models are pre-pruned, limiting the creation of new branches when the node has less than 10 instances.

In the first experiment, we generated five different models using data from the five selected street sections and the five density levels approach. And we tested these models in the five street sections to check accuracy and generality of the learned models. The results for this first configuration are on Table 2.

We can observe that the accuracy is similar for all the street sections except for B, whose behaviour seems to be more difficult to predict. A and E, the two points away from downtown and the Job Center, present a similar behaviour as expected.

In the second experiment, the problem is simplified with only two density levels both for the class and the state predicates. The results for this last configuration are on Table 3.

We can observe that as we decrease the number of density levels, the complexity of the problem decreases too and the prediction task becomes easier. With only two levels, the density of a street knowing the state of the city in the last time

	A	B	C	D	E
A	0.90	0.68	0.85	0.77	0.83
B	0.82	0.72	0.79	0.77	0.80
C	0.83	0.66	0.88	0.77	0.81
D	0.80	0.66	0.83	0.85	0.81
E	0.87	0.66	0.85	0.78	0.89

Table 2: Accuracy results using the model obtained with five density levels. Each cell (i, j) represents the estimated accuracy of learning a model with the data extracted at point i in the city and testing that model against the data collected at point j.

	A	B	C	D	E
A	0.99	0.94	0.99	0.97	0.99
B	0.99	0.95	0.99	0.98	0.99
C	0.96	0.93	0.99	0.97	0.99
D	0.96	0.93	0.99	0.98	0.99
E	0.96	0.93	0.99	0.97	0.99

Table 3: Accuracy results using two density levels for the class and the predicates. Each cell (i, j) represents the estimated accuracy of learning a model with the data extracted at point i in the city and testing that model against the data collected at point j.

steps can be predicted with a high accuracy, even in street sections that have very different behavior. The final model that will be used in our architecture corresponds to the one learned with the data of point B, which on average performs best. The relational tree was shown in Figure 4.

5.2 Results on Traffic Management

Finally, we want to test whether a traffic control system would improve its performance if it had some predictive model of the traffic. To do so, we will use several simulation scenarios where we vary the size of the network (medium and large), the fluency of traffic (fluent or congested) and the evaluated time period (an hour and a day).

When using the learned model, it predicts the density at each street at each time step, using the previous X time steps as input. If it detects a high density at any subset of the street sections, it generates goals to lower the density of those street sections. These new goals, together with the current state of the traffic, create a PDDL planning problem that is given as input to the planner. Therefore, the system is predicting the appearance of goals in the next X time steps, and the planning process can anticipate to the congestions. We will call this new approach `Learning`. In [Pozanco *et al.*, 2016], we show that if the system uses a short-horizon prediction, having the same time steps for both building the model and checking for goals is not that important. So, our system checks for new goals every fifty seconds using the prediction model built with the five minutes time step previously described.

We compare our system with a `Static` one, that corresponds to the default system used by SUMO. We also compare our approach with a `Reactive` system, that acts locally on each traffic light and sets a longer green phase on those whose their corresponding street density is currently high. We also compare with the AP approach proposed in [Gulič *et al.*, 2015], co-winner of the ARTS-COST competition on *Increasing the resilience of road traffic support systems by the use of autonomics*[1]. That planning system does not have any learning component and only calls the planner when a vehicle has been stopped for a long time. We will call it `Planning`. This system is the starting point of our approach, so we use the same planning domain and planner, LAMA [Richter and Westphal, 2010]. The last system we introduce in the tests combines the `Planning` approach and the `Learning` one. It calls the planner when a goal (high density) is predicted or the current density of a street is high. We will refer to it as `Combined`.

We use the following metrics to measure the performance of each system: the number of steps it takes all cars to reach their destination; the total amount of $C0_2$ emitted by the vehicles; the average waiting time (AWT); the average travel time (ATT); and, if it applies, the number of planner executions (PE) and the mean planner execution time (MPE). We choose them simply for comparison, none of the systems explicitly reasons on optimizing these metrics.

Experiments in a Medium-Sized City Network

We created a fluent traffic scenario for the first experiment by introducing 5300 cars in 3600 steps in the same city network we used in the learning goals experiments. The simulation finishes if all cars reach their destination, or after 5000 steps. The results are shown in Table 4. We can see that there is no substantial difference when the traffic is fluid among the different systems. But the `Learning` approach outperforms the others on most metrics. So, when the traffic is fluent, one expects that even the `Static` control program will perform well. In this traffic situation, the time spent on average per vehicle in a traffic light (AWT) is approximately half of the total time spent in their complete travel (ATT). Given the size of the example network, ATT is around three minutes, while AWT is around a minute and a half. The number of planner executions is low in the `Planning` and `Learning` systems, and it becomes very high when using the `Combined` approach. The number of times it calls the planner is much higher than in the two other approaches, as expected.

	Steps	$C0_2$	AWT	ATT	PE	MPE
Static	3969	1103	93	172		
Reactive	4059	1137	100	181		
Planning	4070	1117	95	175	22	10
Learning	3881	1090	88	167	15	10
Combined	4104	1193	115	197	61	10

Table 4: Performance of the different control systems with a fluent traffic situation in a medium-sized city. Steps, AWT and ATT are given in steps (seconds), while $C0_2$ is in kg. MPE is in seconds.

In the second experiment, we test the systems performance on a very congested traffic scenario using the same city network. It was created by introducing 6000 cars in one hour

[1] `https://helios.hud.ac.uk/cost/comp2.php`

(3600 steps). The results are reported in Table 5. The columns report the same metrics as the one before.

	Steps	$C0_2$	AWT	ATT	PE	MPE
Static	-	2553	582	638		
Reactive	4106	1262	119	202		
Planning	-	2187	435	506	48	11
Learning	4070	1265	121	204	46	10
Combined	4244	1301	128	212	68	11

Table 5: Performance of the different control systems with a very congested traffic situation in a medium-sized city. Steps, AWT and ATT are given in steps (seconds), while $C0_2$ is in kg. MPE is in seconds.

As we can see, even if the `Planning` approach outperforms the `Static` system, it performs worse than the `Reactive` mechanism and the two other autonomic approaches. Both `Learning` and `Combined` can completely solve the traffic congestion. The vehicles spend much more time waiting on average than travelling in this scenario (relation between ATT and AWT). However, the `Learning` system is able to reduce the waiting time to half of the travel time, as in a fluent traffic situation. Thus, it is effectively converting a congested situation into a fluent traffic scenario. The reduction of the pollution achieved by `Learning` is quite substantial too: half of the $C0_2$ levels of the static approach. In fact, they are close to those generated in a fluent traffic scenario. `Reactive` obtains practically the same results than the `Learning` approach, even if it only acts locally at each traffic light without considering the whole network.

Dense Traffic in a Large Size City Network

This experiment tests the scalability of the proposed model to larger city networks. The benchmark network in this case is composed of 130 junctions, 520 traffic lights and 566 streets. This can be considered as a large network in relation to most papers in the field, specially considering that our approaches perform centralized planning. The network is shown in Figure 7. We introduce 13,000 cars in one hour in order to create a dense traffic situation. As the city is bigger than the previous one, a experiment will finish when all cars reach their destination or after 6,000 time steps. Table 6 reports the results.

	Steps	$C0_2$	AWT	ATT	PE	MPE
Static	-	6649	439	549		
Reactive	-	7676	605	709		
Planning	-	5520	341	468	50	46
Learning	5837	5231	321	445	47	44
Combined	-	6279	518	633	64	54

Table 6: Performance of the different control systems with a dense traffic situation in a large-sized city network. Steps, AWT and ATT are given in steps (seconds), while $C0_2$ is in kg. MPE is in seconds.

In this case, `Learning` outperforms the rest and it is the only one that can finish the simulation before 6,000 steps. The model we learned with the medium-sized urban network is able to generalize to this larger city. Our system scales quite well even in a large network; it can find a plan in less than fifty seconds, the checking-for-goals sample period. The performance of `Planning` is quite good in this case and it almost solves the congestion. Thus, this only-planning approach works well when we have a reasonably high traffic density (as in this experiment or in the first one), but not too high (as in the previous experiment). The `Reactive` method does not scale up well to the large city network. When trying to locally reduce the congestion, it ends up generating traffic jams and performing even worse than the default, `Static`.

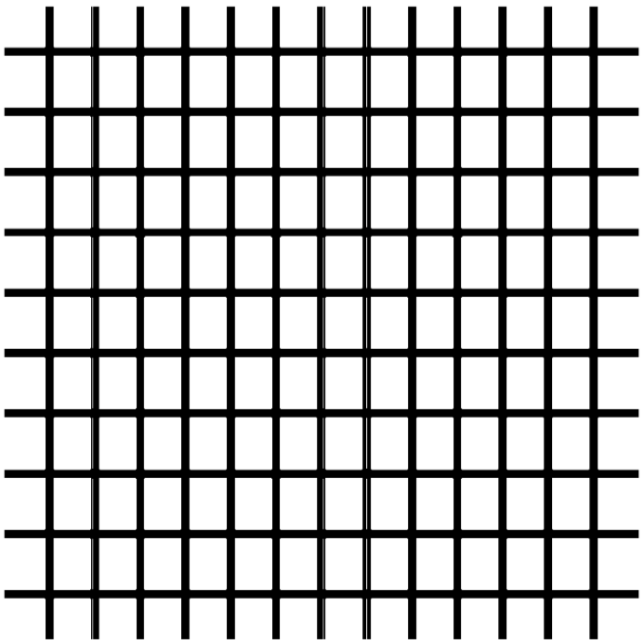

Figure 7: Large city network used in the second type of experiments.

Full day experiment

The last experiment focuses not only on trying to handle a traffic peak, but also to test whether a system can deal with a full day traffic flow. In these cases, the decisions spread over time. We use the medium-sized city network and a traffic demand specification similar to the one presented on Figure 6 for the week days. In this experiment we only measure the AWT per hour. The other metrics could be irrelevant for the 24 hours case. The results are reported on Figure 8. Vehicles routes remain static in SUMO. A car will always try to reach its destination following the shortest path. If this route is congested, the vehicle will not choose another one, but it will stand still waiting for the route to be free. That is the reason why, when using some systems, the network can get congested at some time point and become congested for the whole day. We can see this effect when a given curve in the graphic reaches 200 s. When using this metric, a traffic system performs better if the area under its curve is smaller.

As we can see, only our `Learning` system is able to finish the simulation properly. The AWT grows up in the morn-

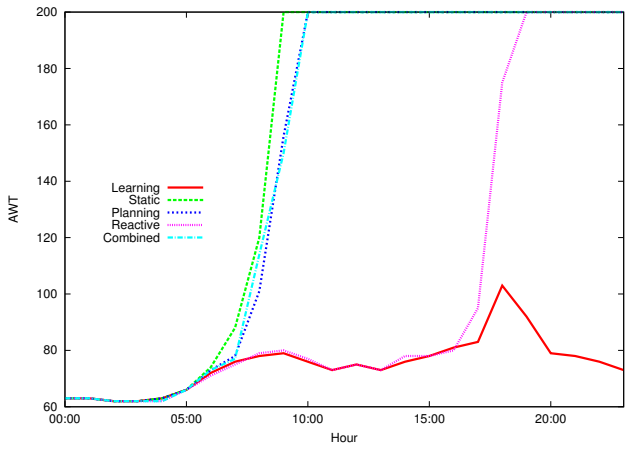

Figure 8: Average waiting time in the city network per hour.

ing, when the cars go to the Job Center, but it does not get fully congested. The AWT remains around 80 s throughout the morning and it starts growing again by the end of the workday. The metric reaches a peak around 18:00 where the AWT is 103 seconds at the most congested traffic situation of the day, which is still a reasonable behavior. After that time period, the system is able to reduce the congestion and the AWT starts to decrease. The `Reactive` system, which showed good performance in the medium-sized city network, can solve the early morning traffic problem. It obtains similar results to the ones of `Learning` until the end of the workday. However, it cannot deal correctly with the end of the day traffic. The other systems can not face the morning rush hour. Even if the `Planning` system is still better than the other two, it does not solve the congestion.

6 Related work

The first UTC models in the 1950s and 1960s, were based on fixed-time traffic lights control mechanisms. Actions were predefined following an off-line optimization using historical data of demand levels. TRANSYT [Robertson, 1969] is one of the most well developed and widely used control systems that uses these techniques. These approaches could even generate "green waves", simple coordination of neighbouring traffic lights in order to increase the traffic fluidity. The problem of early systems is that they can age rapidly due to the continuous evolution of the traffic flows in a city. The benefits may be lost in some years if the control plans are not updated. Our proposed system overcomes this situation, as it not only can react to the current traffic scenario, but it can anticipate and adapt to future ones.

In the last years, the use of new and better sensor systems has allowed engineers to implement traffic-responsive systems that use the data provided by the detectors in an on-line way. These techniques range from centralized approaches, as SCOOT [Bretherton *et al.*, 1998] and SCATS [Lowrie, 1990] to distributed ones as UTOPIA[Donati *et al.*, 1984]. As most other traffic-responsive systems, they use a mathematical framework to compute the optimal time allocation of each traffic light. A weak point of these systems is that they cannot predict incidents and they do not deal well with them. Also, their models are not defined declaratively. Thus, our models are easier to update with new types of information, or new metrics to be taken into account when optimizing.

Other AI-related approaches have appeared in recent years. The main goal is to build semi- or fully autonomous systems with little human assistance. Most of them address traffic management from a multi-agent perspective. A single agent acts over a single junction or subset of junctions and then several agents collaborate, discuss and negotiate with the rest [Ossowski *et al.*, 1998]. In [Box and Waterson, 2012], the authors propose a model based on logistic regression and neural networks to learn over time how to better control the traffic signals. Other approaches focus on multi-agent reinforcement learning [Kuyer *et al.*, 2008], distributed geometric fuzzy systems [Gokulan and Srinivasan, 2010] or creating a multi-agent model predictive control [de Oliveira and Camponogara, 2010]. New approaches for efficient UTC are arising in the last years using vehicle communication as the core of the control process [Ferreira *et al.*, 2010]. But, these methods are still far from being implemented in real cities and controlling traffic lights remains the most widespread way to handle urban traffic.

7 Conclusions and Future work

In this paper we have presented a dynamic approach for UTC based on Automated Planning and Relational Learning. As we have shown, by adding a learning component that can predict the city state to a planning system, we can highly increase its autonomy. It can automatically generate its own goals, in addition to letting the planner starts the planning process sooner. We have tested our model in several traffic control scenarios, showing that the ability to anticipate goals can lead to better control performance than using only static traffic lights programns. Our system also outperforms the `Planning` system and overcomes its limitations, as `Planning` needs to know when a vehicle has been stopped for a long time. Instead, our model only needs the street density levels, which are easier to obtain from current sensor systems. By just knowing density levels, we are able to model a wide variety of circumstances that affect traffic behavior such as adverse weather conditions or different days and hours. Also, since other types of incidents (e.g., road-blocking or big accidents) indirectly affect the density levels, we believe our approach could also work to alleviate congestions caused by them.

In future work, we would like to integrate the ability to learn how to anticipate goals with externally supplied goals (e.g., by traffic controllers), reactively generated ones (e.g., reactively generating goals), or internally supplied ones (e.g., generated by internal motivations of the system). Although the proposed system scales up, we would also like to apply a multi-agent approach by dividing the city in sections in which an agent can apply the system in an autonomous way. We think this could lead to similar performance with lower execution times. We would also like to compare our system with other state of the art methods on traffic control, such as

model predictive control (e.g., SCOOT), or other AI-based approaches (e.g., reinforcement learning). Finally, we want to test the proposed system in irregular city networks such as European ones and build the learning model on-line in order to show the system's real-world applicability.

Acknowledgements

This work has been partially supported by MINECO project TIN2014-55637-C2-1-R.

References

[Behrisch *et al.*, 2011] Michael Behrisch, Laura Bieker, Jakob Erdmann, and Daniel Krajzewicz. Sumo–simulation of urban mobility. In *The Third International Conference on Advances in System Simulation (SIMUL 2011), Barcelona, Spain*, 2011.

[Blockeel and De Raedt, 1998] Hendrik Blockeel and Luc De Raedt. Top-down induction of first-order logical decision trees. *Artificial intelligence*, 101(1):285–297, 1998.

[Bonet and Geffner, 2005] Blai Bonet and Héctor Geffner. mGPT: A probabilistic planner based on heuristic search. *JAIR*, 24:933–944, 12 2005.

[Box and Waterson, 2012] Simon Box and Ben Waterson. An automated signalized junction controller that learns strategies from a human expert. *Engineering applications of artificial intelligence*, 25(1):107–118, 2012.

[Bretherton *et al.*, 1998] R. Bretherton, K. Wood, and G.T. Bowen. Scoot version 4. In *Proceedings of 9th International Conference on Road Transport Information and Control*, 1998.

[de Oliveira and Camponogara, 2010] Lucas Barcelos de Oliveira and Eduardo Camponogara. Multi-agent model predictive control of signaling split in urban traffic networks. *Transportation Research Part C: Emerging Technologies*, 18(1):120–139, 2010.

[Donati *et al.*, 1984] F Donati, Vito Mauro, G Roncolini, and M Vallauri. A hierarchical decentralized traffic light control system. the first realisation "progetto torino". In *Proceedings of the 9th World Congress of the International Federation of Automotive Control*, pages 2853–2858, 1984.

[Ferreira *et al.*, 2010] Michel Ferreira, Ricardo Fernandes, Hugo Conceição, Wantanee Viriyasitavat, and Ozan K Tonguz. Self-organized traffic control. In *Proceedings of the seventh ACM international workshop on VehiculAr InterNETworking*, pages 85–90. ACM, 2010.

[Fox and Long, 2003] Maria Fox and Derek Long. PDDL2.1: An extension to PDDL for expressing temporal planning domains. *Journal of AI Research*, 20:61–124, 2003.

[Gokulan and Srinivasan, 2010] Balaji Parasumanna Gokulan and Dipti Srinivasan. Distributed geometric fuzzy multiagent urban traffic signal control. *Intelligent Transportation Systems, IEEE Transactions on*, 11(3):714–727, 2010.

[Gulič *et al.*, 2015] Matija Gulič, Ricardo Olivares, and Daniel Borrajo. Using automated planning for traffic signals control. In *Working Notes of ARTS-COST 2nd competition*, 2015.

[Guzmán *et al.*, 2012] César Guzmán, Vidal Alcázar, David Prior, Eva Onaindía, Daniel Borrajo, Juan Fdez-Olivares, and Ezequiel Quintero. PELEA: a domain-independent architecture for planning, execution and learning. In *Proceedings of ICAPS'12 Scheduling and Planning Applications woRKshop (SPARK)*, pages 38–45, Atibaia (Brazil), 2012. AAAI Press.

[Hamilton *et al.*, 2013] Andrew Hamilton, Ben Waterson, Tom Cherrett, Andrew Robinson, and Ian Snell. The evolution of urban traffic control: changing policy and technology. *Transportation planning and technology*, 36(1):24–43, 2013.

[Kuyer *et al.*, 2008] Lior Kuyer, Shimon Whiteson, Bram Bakker, and Nikos Vlassis. Multiagent reinforcement learning for urban traffic control using coordination graphs. In *Machine learning and knowledge discovery in databases*, pages 656–671. Springer, 2008.

[Lowrie, 1990] PR Lowrie. Scats, sydney co-ordinated adaptive traffic system: A traffic responsive method of controlling urban traffic. 1990.

[Ossowski *et al.*, 1998] Sascha Ossowski, José Cuena, and Ana García-Serrano. A case of multiagent decision support: Using autonomous agents for urban traffic control. In *Progress in Artificial Intelligence—IBERAMIA 98*, pages 100–111. Springer, 1998.

[Papageorgiou *et al.*, 2007] M Papageorgiou, M Ben-Akiva, Jon Bottom, Piet HL Bovy, SP Hoogendoorn, Nick B Hounsell, Apostolos Kotsialos, and M McDonald. Its and traffic management. *Handbooks in Operations Research and Management Science*, 14:715–774, 2007.

[Pozanco *et al.*, 2016] Alberto Pozanco, Susana Fernández, and Daniel Borrajo. On learning planning goals for traffic control. In *4th Workshop on Goal Reasoning (IJCAI'16)*, 2016.

[Richter and Westphal, 2010] Silvia Richter and Matthias Westphal. The lama planner: Guiding cost-based anytime planning with landmarks. *Journal of Artificial Intelligence Research*, 39(1):127–177, 2010.

[Robertson, 1969] Dennis I Robertson. Transyt: a traffic network study tool. 1969.

[Vallati *et al.*, 2016] M. Vallati, D. Magazzeni, B. De Schutter, L. Chrpa, and T.L. McCluskey. Efficient macroscopic urban traffic models for reducing congestion: a pddl+ planning approach. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence (AAAI-16)*, 2016.

[Yoon *et al.*, 2007] Sungwook Yoon, Alan Fern, and Robert Givan. FF-replan: A baseline for probabilistic planning. In *ICAPS*, pages 352–360, 2007.