On-the-Fly Knowledge Acquisition for Automated Planning Applications: Challenges and Lessons Learnt

Saumya Bhatnagar[1][a], Sumit Mund[1], Enrico Scala[2][b], Keith McCabe[3], Thomas L. McCluskey[1][c] and Mauro Vallati[1][d]

[1] *School of Computing and Engineering, University of Huddersfield, U.K.*

[2] *University of Brescia, Italy*

[3] *Simplifai Systems Limited, U.K.*

Keywords: Automated Planning, Knowledge Acquisition, Traffic Control.

Abstract: Automated planning is a prominent AI challenge, and it is now exploited in a range of real-world applications. There are three crucial aspects of automated planning: the planning engine, the domain model, and the problem instance. While the planning engine and the domain model can be engineered and optimised offline, in many applications there is the need to generate problem instances on the fly. In this paper we focus on the challenges of on-the-fly knowledge acquisition for complex and variegated problem instances. We consider as a case study the application of planning to urban traffic control and we describe the designed and developed knowledge acquisition process. This allows us to discuss a range of lessons learned from the experience, and to point to important lines of research to support the knowledge acquisition process for automated planning applications.

1 INTRODUCTION

Automated planning is one of the most prominent AI challenges; it has been studied extensively for several decades, and it is now exploited in a wide range of applications. Examples include network security penetration testing (Hoffmann, 2015), battery load management (Fox et al., 2012), train dispatching (Cardellini et al., 2021) and control of robots (Kvarnström and Doherty, 2010; Capitanelli et al., 2018).

The three aspects in automated plan generation focused on in the AI literature are the planning engine itself, the domain model that captures the physics of the problem area, and the problem instance that contains a specification of the initial status of things, the relevant objects and the goals to be pursued. Planning engines used in real-world applications rely on the ability that utilising the operational semantics of the domain model will faithfully simulate progression over time of the state of the real world (in particular, crucial in the validation of any generated plan). Not only has the domain model to be an accurate representation of the application's dynamics for this purpose, but the problem instance has to adequately and accurately reflect the current state of the world, and the goal required to be solved. Knowledge engineering and knowledge acquisition issues of the mentioned models and instances are exacerbated in real-time AI planning applications, where the problem instance must be acquired on the fly to allow an agent to deal with problems as they occur. Beside knowledge engineering issues related to the engineering of the domain model that have received significant coverage (McCluskey and Porteous, 1997; McCluskey et al., 2017; Vallati and McCluskey, 2021), there is the critical aspect of generating problem instances on the fly. This is also important taking into account how the poor quality of problem instances can affect the ability of state-of-the-art planning engines to solve the problem at hand (Vallati and Chrpa, 2019). Indeed, automation in the construction of a complex initial state not only aids the quality of the problem instance, but helps in the efficiency of knowledge capture. The latter is essential for the cost-effectiveness of employing automated planning in applications. Early work on such automation was demonstrated in the nine contestants of the International Competition on Knowledge Engineering for Planning and Scheduling (ICKEPS) 2009

[a] https://orcid.org/0000-0002-3425-3394
[b] https://orcid.org/0000-0003-2274-875X
[c] https://orcid.org/0000-0001-8181-8127
[d] https://orcid.org/0000-0002-8429-3570

(Barták et al., 2010), which focused on automation in the generation of planning knowledge from existing application-held data structures, and by recent works on using templates to generate instances (Long et al., 2018; Gregory, 2020).

In this paper we focus on aspects of knowledge acquisition in real-time applications where the initial state is complex and variegated, consisting of (i) persistent or static structures, and (ii) a set of values that must be acquired and processed on the fly. We consider as a case study the application of AI planning to urban traffic control (McCluskey and Vallati, 2017) for generating in real time traffic signal strategies for a major road corridor of the Kirklees council, that is situated in the Yorkshire county of the United Kingdom. We describe the knowledge acquisition process that has been designed and developed, and we take the opportunity to provide insights into the challenges of generating, validating, and verifying complex initial states on the fly. In particular, we discuss the lessons learnt, and we point to important lines of research for knowledge engineering and acquisition to foster the use of AI planning in real-world applications.

2 RESEARCH CONTEXT

Traditional approaches to urban traffic control are based on the idea of generating fixed strategies for frequent traffic patterns, such as morning and evening peaks, and off peaks. This approach is problematic when unusual or unexpected events happen: considering the impact of COVID-19, for instance, traffic volumes have suddenly varied from -80% to $+30\%$ of typical pre-COVID traffic conditions, and the composition and journeys of traffic has drastically changed as well. To deal with such quickly-changing conditions, strategies of interventions have to be generated on the fly, considering the current actual conditions of the network.

Generating a detailed strategy of interventions to manage an unusual situation *in real time* is considered to be beyond the capacity of human operators. In this situation, automated planning can help strategy generation if data describing the current situation is available and adequate, and a domain model has been constructed and validated to mirror the application domain.

In this paper we consider an urban road traffic management scenario, where strategies generated are changes to traffic signal timings over a period of time, in response to some real-time goal. Transport operators need the ability to produce regional strategies in real time which will deal with abnormal or unexpected events such as road closures and incidents. These cause huge delays and decreased air quality because of excessive congestion and stationary traffic. The existing conditions and set of corrective goals required to deal with these events are so varied that detailed strategies are impossible to draw up a priori in a large, dense urban area.

There is a growing interest in the planning and scheduling community in dealing with urban traffic control problems, particularly with regards to traffic signal control. On the scheduling side, the SURTRAC approach utilises a distributed scheduling system which controls traffic signals in urban areas (Xie et al., 2012; Hu and Smith, 2019; Smith, 2020). In SURTRAC, each junction is controlled by a scheduling agent that communicates with connected neighbours to predict future traffic demand, and to minimise predicted vehicles waiting time at the traffic signal. It is currently deployed in Pittsburgh, USA, with its distributed approach suggesting good scale-up but less goal flexibility than if utilising a centralised AI planning approach.

On the planning side, Gulic et al's system (Gulić et al., 2016) involves joining together a SUMO simulator (Lopez et al., 2018) to an AI Planner, via a monitoring and execution module called the "Intelligent Autonomic System". The planning representation was done using PDDL 2.1 (Fox and Long, 2003), with no explicit representation of vehicles in the planner. Instead, traffic concentrations on road links are represented by relative density descriptors, such as very-low, low, medium and high. Traffic light change actions are enumerated to cover all the ways that a particular configuration would effect the arrangements of road links. By abstracting away from explicit counts of vehicles, the system can deal with regions containing thousands of vehicles. Also, the close coupling with SUMO demonstrates the use of monitoring and replanning very effectively, and allows exhaustive testing of the system under sets of disturbances (vehicle influx, road closures). The work by Pozanco et al. (Pozanco et al., 2021) exploits a similar approach to those of Gulic et al, but extends it in a number of ways: the most remarkable being the ability to learn and continuously evolve the knowledge model to better adapt to the network behaviour. A different line of work (Vallati et al., 2016; McCluskey and Vallati, 2017) exploits PDDL+ for encoding a flow model of vehicles through traffic-light controlled junctions. The length of traffic light phases are under the control of the planner, that can decide to prioritise some traffic flows, in order to reach specified goals (a phase determines which of the flows through that junction are on and have traffic flowing). Goals are specified in terms of numbers of vehicles desired on some critical road links. This work follows on the PDDL+ approach. Next section is devoted to explain what a PDDL+ planning problem is, and what entails. Then, we see the PDDL+ model of the urban traffic control and how the various pieces of the encoded knowledge are updated in an online, real-time context.

3 THE CASE STUDY

In this section we briefly introduce the PDDL+ language, and the designed problem model, and then describe the characteristics of the target urban region. Here we build on the work done in (McCluskey and Vallati, 2017) to use PDDL+ planning to generate strategies for dealing with unexpected or abnormal circumstances in a controlled urban region, such as accidents, roadworks, etc.

3.1 PDDL+ Planning

Automated planning deals with the problem of finding a plan (a sequence of actions) that transforms the environment from an initial state to some desired goal state (Ghallab et al., 2004).

A planning task comprises a domain model and a problem model, where the former defines operators while the latter consists of objects, initial state and goal condition. The domain model and the problem model are referred to as the *knowledge models*, as they encode all the knowledge needed by an automated reasoner to solve the corresponding task.

PDDL+ is a language based on first-order logic that uses propositional, numeric predicates called fluents together with the standard connectives used in logic to postulate boolean and numeric formulas over them.

A PDDL+ domain model is defined by the tuple $\langle F, X, A, E, P \rangle$:

- F and X are sets of propositional and numeric fluents, respectively. A propositional fluent can be true or false. A numeric fluent can instead take any value from $Q \cup \{\bot\}$.
- A is the set of actions. Each action is a pair $a = (pre(a), eff(a))$ where $pre(a)$ is a first order formula, and $eff(a)$ is a set of Boolean and numeric effects. Boolean effects are assignments $\langle p, \{\top, \bot\} \rangle$ with $p \in F$ where numeric effects are assignments $\langle p, \xi \rangle$, with ξ being an arithmetical expression.
- P is the set of processes. As an action, a process p is a pair $p = (pre(p), eff(p))$ with the difference being that $eff(p)$ only involves numeric additive assignments.
- E is a set of events. Events are syntactically equivalent to actions.

The difference between actions, and processes and events is that, while actions are under the control of the agent, processes and events are spontaneous changes of the environment that apply as soon as their precondition holds.

Given a PDDL+ domain model, a PDDL+ problem model is the tuple $\langle I, G \rangle$ where:

- I is a complete fluent assignment over F and X representing the *initial state* .
- G is syntactically equivalent to a precondition of an action, and represents the *goal condition*.

As an innovation w.r.t. more basic forms of planning (Ghallab et al., 2004), a PDDL+ plan is a set of actions from A associated with a time-stamp, and a time horizon. Intuitively, the time-stamp indicates exactly when an action is to be executed. A plan is said to be valid *iff* all actions are applicable at their time and the goal is satisfied at the prescribed horizon. More details about the syntax and semantics of PDDL+ can be found in (Fox and Long, 2006).

3.2 The Problem Model in Urban Traffic Control

A region of the *road network* can be represented by a directed graph, where edges stand for *road links* and vertices stand for *junctions*. One vertex is used for representing the *outside* of the modelled region. Intuitively, vehicles enter (leave) the network from road sections connected with the outside. Each link has a given maximum *capacity*, i.e. the maximum number of vehicles that can be, at the same time, in the corresponding road, and the current number of vehicles of a road link, which is denoted as *occupancy*.

Traffic in junctions is distributed by *turnrates* that are defined using a dedicated predicate, between couples of road links. Given two links r_x, r_y, a junction i, and a traffic signal stage p such that r_x is an incoming link to the junction i, r_y is an outgoing link from i, and the flow is active (i.e., has green light) during stage p, the flow rate (r_x,r_y,i,p) stands for the maximum number of vehicles that can leave r_x, pass through i and enter r_y per time unit. Notably, turn rates are defined only for permitted traffic movements. In other words, the existence of the predicate gives information about the structure and topology of the network, while its numeric value provides information about the amount of vehicles that can move. In our model, the turn rate value is given in “passenger car units” PCUs per second. For the sake of simplicity, we assume that vehicles going in the same direction move into the correct lane, thus not blocking other vehicles going in the different directions.

Junctions are described in terms of a sequence of traffic signal stages. Specifically, junctions *contain* a signal stage, and stages are connected using a *next* predicate to define their sequence. According to the active traffic stage, one (or more) flow rates are activated, corresponding to the traffic lights that are turned green. For each phase, the *minimum* and *maximum* stage length is specified. Within this range, the planner can decide whether to stop the phase currently active, or not. Between two subsequent signal stage, an *intergreen* interval is specified. Intergreens are (usually) short periods of time designed to allow vehicles that are stacked in the middle of the junction to leave, and pedestrian crossing time, before the next stage is started. The state of a junction is specified using an *active* predicate, that encodes which stage a junction is on, and the *greentime* and *intertime* values, that specify either the greentime of the currently active stage, or how much time has been spent in the intergreen between the active stage and the next one. The model was encoded so that some junctions can be declared as not under the control of the planner, by introducing an artificial predicate called *controllable*.

Given a fully specified traffic planning problem, and using a domain model based on that described in (McCluskey and Vallati, 2017), the goal is currently specified in terms of desired occupancy of some road links. The task is to obtain this as soon as possible by generating a traffic signal strategy that optimises the length of traffic stages on the controlled junctions.

Figure 1 shows an excerpt of a problem model, where the initial state of a junction `j1` is specified: `stage1` is currently active and has been on green for 4 seconds; `stage1` is followed by `stage2`, and the green time of `stage1` ranges between 10 and 120 seconds. When `stage1` is on green, two traffic flows are active: one from `road1` to `roadX`, and one between `road2` and `roadX`. The occupancy and capacity of `road1` is also specified. Finally, a fictional goal is described as the required occupancy of one of the links of the network.

3.3 The Urban Region

The modelled area is situated in West Yorkshire, United Kingdom, specifically within the Kirklees council. It consists of a major corridor that links the Huddersfield ring road with the M1 highway and the southern part of the Kirklees council. It is heavily used by commuters and by delivery vans to get to the centre of the Huddersfield town, or to move between the M62 and the M1 highways. The corridor is approximately 1.3 kilometres long, and consists of 6 junctions and 34 road links. Each junction has between 4 and 6 stages, and between 10 and 17 valid traffic movements. A schema of the considered region is shown in Figure 2, in terms of links, junctions, and connections with the outside region.

Traditionally, AI approaches for signal traffic control relies on traffic models, where the approaches can be tested and optimised. The AI-based approach interacts with a traffic simulator that provides all the required data and performs all the simulation-related

```
(:init
(active stage1)
(= (greentime j1) 04)
(= (intertime j1) 00)
...
(contains j1 stage1)
(contains j1 stage2)
(next stage1 stage2)
(= (intergreen stage1 stage2) 05)
(controllable j1)
...
(= (turnrate stage1 road1 roadX) 00.3)
(= (turnrate stage1 road2 roadX) 00.2)
...
(= (mingreentime stage1 ) 10 )
(= (maxgreentime stage1 ) 120 )
...
(= (occupancy road1) 45.0)
(= (capacity road1)   54.0)
...
)
(:goal
(< (occupancy road1) 20.0)
)
```

Figure 1: An excerpt of a PDDL+ problem model focusing on a single junction `j1`, and partially describing three links.

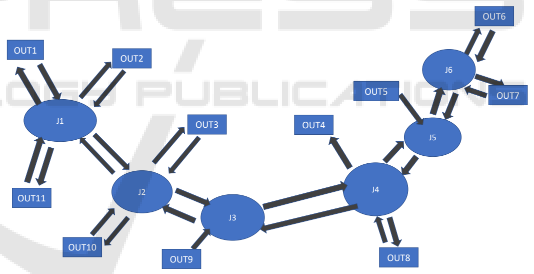

Figure 2: A simplified overview of the modelled corridor, in terms of junctions, links, and connection with the outside region (rectangles).

tasks. This is also the case of previous work exploiting AI planning approaches, where AIMSUN or SUMO models were used (Vallati et al., 2016; McCluskey and Vallati, 2017). However, for the considered area, and for most traffic networks, there is no such traffic model available. This poses a significant challenge for the exploitation of AI planning approaches, both in terms of data collection and in terms of validation of the generated strategies. Notably, the PDDL+ knowledge model can be also used as a simulator, since it captures the dynamics of the movement of traffic, as long as we can collect data to be used as state information within PDDL+. To do this we note that large parts of urban traffic systems are already controlled by sensor-rich traffic-responsive mechanisms. In our region, all the junctions of the area are controlled using SCOOT (Taale et al., 1998), and we can leverage on SCOOT sensors to collect data that is needed by the AI planning system. SCOOT is a demand driven, traffic responsive control aimed at handling cycle-to-cycle changes in demand. In response to changes in traffic flows, SCOOT would gradually adapt and adjust the traffic signal timings. SCOOT is dependent on its own local data sensors, usually inductive loops embedded in the road surface. Further, SCOOT stores the data coming from its sensors, and its internal behaviour, into a dedicated database called ASTRID (Hounsell and McDonald, 1990). The data stored into ASTRID become a valuable source of knowledge that can be used to automatically extract information about the structure of the controlled region, as well as its recorded condition. However, ASTRID is used to access historical data, as it requires several minutes to process the data received by SCOOT before making it available. There is therefore the need to get access to SCOOT sensors to obtain a real-time understanding of the traffic conditions.

4 ON-THE-FLY INSTANCE GENERATION

For the considered case study, the planning system would be invoked when unusual circumstances are recorded in the controlled region. When this is the case, there is the need to generate on the fly a planning problem instance that accurately describes the traffic network, and its current condition. It is pivotal to generate instances on the fly also because the quality and informativeness of data, particularly with regards to traffic flows, decay very quickly over time: to contextualise this aspect, consider that the traffic demand in 15 minutes will include vehicles that, at the current time, are tens of kilometres away from the controlled region – travelling via the nearby motorway. There is a broad range of events that can affect traffic flows in such a large spatio-temporal space.

For designing and developing the on-the-fly knowledge acquisition process, we took a systematic approach. Starting from the problem model specification (as shown in Figure 1), data required for the initial state description have been classified according to their static or dynamic nature. The former refers to data that do not change within the class of problems that it is currently addressed (in other words, does not change when the considered urban region remains the same). The latter refers to elements that continuously change according to the status of the network and of its junctions and links. This discrimination is made in terms of PDDL+ predicates of the initial state description. For each required PDDL+ predicate, the relevant data sources have been identified, its format specified, and the flow from raw data to the PDDL+ predicate has been defined and documented. Static data is extracted once per geographical region, via a dedicated pipeline, thoroughly validated and then stored in an appropriately structured knowledge base. Dynamic data instead have to be extracted on-demand, and dedicated adaptors and processing steps have to be designed and deployed. Further, where possible, validation and verification approaches have been put in operation – to take into account faulty data or flaws in the data flow.

Considering the case study problem, the following data can be considered as static for a urban region to control:

1. The topology of the road links, junctions, and region boundaries. This is encoded implicitly in our model: the `turnrate` predicates (their existence) are used to connect links and junctions, and each link, junction, and stage corresponds to a PDDL+ object.
2. The vehicle capacity of all the road links. In our model, this is given in numbers of PCUs which takes into account the differing size of vehicles using the `capacity` predicate.
3. The specification of the stages of signals, stage progression constraints and the minimum and maximum time that a signal stage can be set for. This is encoded using a list of predicates such as `contains`, `next`, `mingreentime`, and `maxgreentime`.
4. Intergreen timings. Their duration is dependent on stages (preceding and succeeding) and junction. This is encoded using a dedicated `intergreentime` predicate.

5. The permitted traffic movements in a junction, i.e. for each incoming link r_i to a junction, all the destination links that can receive traffic from r_i via the considered junction. These are specified by the existence of a `turnrate` predicate for each possible movement.

A large amount of data is instead dynamic, and need to be adapted and processed according to the day and time that the considered problem is modelling.

A. The average traffic flows between links in number of PCU's per second. This information represents the number of vehicles flowing via a particular traffic movement of a junction at the considered time of day, when the corresponding traffic signal stage is green. Flows in and out of boundary junctions are a special case of this. As far as it is concerned by the PDDL+ language, this kind of data is encoded by the numeric value of `turnrate` predicates.

B. The occupancy of the links of the network at the initial considered time, i.e. the number of vehicles for each of the links, expressed in PCUs. This is encoded using a list of `occupancy` predicates.

C. The state of all the junctions, in terms of stage currently active (or intergreen) and time spent in that state. For each junction, this is represented using the `active`, `greentime`, and `intertime` predicates.

Further, there is another kind of data that can possibly be required: the way in which the unusual circumstances are affecting the controlled region. For instance, in the case of a road traffic accident, a lane may operate at reduced capacity, affecting the incoming and outgoing traffic flows. This kind of data can be labelled as dynamic-unpredictable, since it is impossible to accurately predict, and it is strongly dependent on the current circumstances.

Figure 3 provides an overview of the framework that has been designed to automatically generate a problem instance, green arrows indicate necessary input. This comes under the form of (i) SCOOT data being generated in real-time and stored into the ASTRID database; (ii) specification of links, junctions, etc. not under the direct control of SCOOT and therefore not included in ASTRID; (iii) a manual specification of the goal to be achieved; (iv) manual specification of the changes to the model due to the unexpected conditions (if needed), and (v) a specification of the considered date and initial time. Date and time are needed to correctly define the initial state of the problem, particularly with regard to dynamic data.

The ASTRID database represents the cornerstone of the architecture, as it stores all the data generated and sensed by the SCOOT system deployed in the region. From ASTRID, a number of reports, under the form of structured files, are extracted and processed, in order to produce part of the data required for the initial state description of the planning problem. Notably, ASTRID is not suitable to be used for real-time data, as it requires several minutes to process the SCOOT input and make it available. Further, ASTRID is not the only source of data about the structure of the network. There is usually a number of links and junctions not under the direct control of a SCOOT system, that are therefore missing. This type of additional data can come under different forms: for the considered case study, it was extracted by manually checking maps of the region. Static data of type 1–5 is generated once for a considered urban region. We created parsers which extract static network information from fixed format ASTRID reports which include junctions, links, legal traffic movements, etc. Such static data is then also needed to calculate dynamic traffic flows (as in A. above) for a specified day and time. This is to be done by considering historical data for similar period of the year, time of the week, and time of the day. Finally, dynamic data about link occupancy and state of traffic lights (B. and C. above) is generated by processing the direct output provided by SCOOT, on the basis of the known structure that is derived by ASTRID.

Crucially, the time intervals over which dynamic data are made available and can be collected (or recorded in ASTRID) greatly varies according to the type of data, and to the circumstances. For instance, considering type C, for a given junction the SCOOT system provides an update only when a stage ends, and not at regular intervals. This means that it is not possible to know a-priori when the next update will come, as it depends on when the SCOOT system decides to end the stage that is currently active. Similarly, information about the occupancy of a link (type B) are updated only a moment before the considered link is allowed to discharge, i.e. a traffic light that allows traffic to leave the link is set to green.

5 DISCUSSION AND LESSONS LEARNT

This section focuses on the main lessons learned from the challenges we had to overcome, and those still outstanding.

Complexity of the Acquisition Process. Previous applications of AI planning to urban traffic control relied on the use of a traffic simulator as a proxy for the real world. This greatly simplifies the knowl-

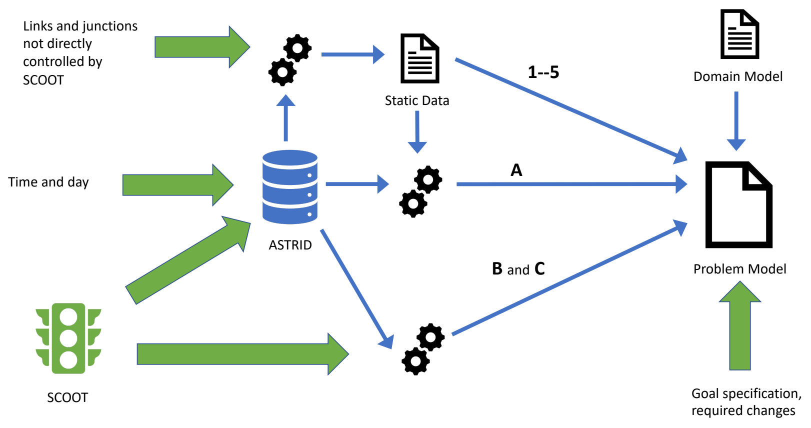
Links and junctions not directly controlled by SCOOT
Static Data
1--5
Domain Model
Time and day
A
ASTRID
B and C
Problem Model
SCOOT
Goal specification, required changes

Figure 3: Overview of the knowledge acquisition process for generating a planning instance for the case study. Green arrows indicate input to be provided. Numbers and capital letters refer to the corresponding type of static and dynamic data.

edge acquisition process: all the elements are already named using a unique identifier, data are consistent, and all the units of measurement are unified and coherent. That is usually not the case when planning knowledge models have to be generated on-the-fly from a multitude of different data sources, including historical data, real-time sensors, etc. As we early discovered when modelling the case study area, the same entities can be named in different ways according to: (i) the data source, and (ii) the expected use of the corresponding bit of information. There is also no guarantee that data stored in different databases are consistent and correct when pulled together, and that the same measurement units are used. In practice, the above means that there is the need to: (i) fully assess and understand all the data sources to be able to grasp the differences; (ii) design and develop dedicated interfaces for each source to extract and format data; (iii) design and develop approaches to merge data pulled from different sources, and (iv) thoroughly validate and verify that once merged, the resulting knowledge is correct and operational. In the absence of a unified model that encompasses all the data sources, the final validation and verification step is extremely cumbersome, and may require manual validation and verification. In our case study, we had to resort to manual verification and debugging of static data (as for Figure 3).

Data Interpretation. Data pulled from different sources may require different interpretations. This is not directly connected to the inconsistent use of identifiers or measurement units, but more related to the semantics of the data. The correct interpretation is not explicitly described in the database or in the pulled file / report, as it is not needed by domain experts, but is instead described in dedicated documentation. This documentation is written for domain experts (road traffic operatives), and heavily relies on domain-specific acronyms and concepts. While in many cases there is a single semantically and syntactically correct interpretation, there are cases where the syntax of the data structure can lend itself to multiple interpretations – those cases are the more challenging to deal with, as the fact that an incorrect interpretation is used can be hard to spot. In our case study we faced this issue with one of the reports generated by the ASTRID system. The report provides the sequence of one type of model message (out of 94 types of model messages) under the name of M37 messages, that are generated by SCOOT to record traffic light stages duration. Such reports have a single line per message, and its syntax supports two alternative interpretations: one where a message describes what is going to happen next, and the other where a message retrospectively describes what just happened. Selecting the wrong interpretation can hinder the exploitation of the overall planning-based traffic control approach, as initial condition of the planning problem will not accurately reflect the real-world status.

Verification of the Initial State. In recent years, there has been a growing interest within the planning community in tools and techniques for supporting the design and deployment of planning techniques in complex real-world applications (Vallati and Kitchin, 2020). This resulted in tools such as VS-studio (Long et al., 2018) and approaches that rely on templates to generate initial state descriptions of planning problems (Gregory, 2020). However, most of the existing tools do not provide appropriate support for the verification of the initial state of a planning problem. This issue can have a limited impact when less expressive planning formalisms (such as classical or numeric planning) are used, but becomes pivotal for PDDL+. Existing tools for supporting the knowledge acquisition of PDDL+ initial state descriptions mostly rely on templates for automatising the process given a valid, correct and consistent knowledge base from which the initial state can be derived. There is a lack of approaches that, given a PDDL+ domain and problem models, can help to verify whether the provided initial state description is syntactically correct, but semantically wrong for the application domain. In the considered case study, examples of this include cases where the maximum green time is lower than the minimum for a given stage, some of the traffic flows have negative values, or the occupancy of a link is higher than its capacity. Due to the semantics of PDDL+, such issues are hard to debug, as they may not prevent a planning engine from generating plans or from parsing the model. To be automatically fixed, they require dedicated techniques to be developed, either based on the knowledge encoded in the domain model, or based on some additional domain-specific knowledge provided aside from the PDDL+ models.

Goal Definition. In the urban traffic control domain, there are both qualitative and quantitative ways to define desirable conditions of a traffic network. However, it is not straightforward to translate them into an AI planning goal definition, as required by the used PDDL+ language. On the one hand, the general idea of having a goal to reach suits the application domain, as the planning-based approach is expected to be utilised when unexpected or unusual issues arise; this kind of exploitation supports the definition of a goal to be reached to mitigate or solve the detrimental effects of the issue(s). On the other hand, in many cases there is not a direct translation between the traffic engineering "goal" conditions, and a PDDL+ formula that describes the properties of the desired status of the network. In its current implementation, the goal definition is left to be manually specified. A step towards a fully automated on-the-fly generation of problem models is the use of predefined goal templates: considering a range of expected issues, templates of goal definitions can be designed. On the fly, an appropriate template can be selected and populated according to the characteristics of the planning problem at hand. This will require the integration of dedicated techniques for guaranteeing that the goal is reachable and it fits the needs of the network conditions – ensuring that it will not lead to a worsening of the issues.

Handling Time. The generation of the initial state using the described data flow is extremely quick, and is usually completed in less than 1 CPU-time second. On the other hand, the planning process can take up to a few minutes: in our case study we set the cutoff time to 5 minutes, but plans are usually generated in less than 1 minute. The time needed to generate a strategy plan means that, by the time the plan is ready the conditions of the network and of the junctions have changed, and the plan is no longer applicable. To overcome this problem we exploit the simulation capabilities of the PDDL+ model: simulating the evolution is the traffic conditions is extremely fast, and can be done considering historical data from a similar time period. Given the initial conditions, we simulate the evolution of the network for 1 minute, and we then use the resulting state as the initial state of the planning process. In this way, we can preserve the applicability of the generated plans, and allow up to 1 minute for the planning process. If a plan is not generated by the end of the 1-minute time window, we can repeat the process and reduce the planning horizon – i.e. the temporal length of the generated strategy – to simplify the search process. We usually consider short planning horizons of at most 15 minutes. In other words, we aim at generating a traffic light strategy that covers the following 15 minutes at most. This is also done to reduce the probability that the network conditions diverge from the simulated ones. Comparing the data obtained from the PDDL+ simulator run on a SCOOT-generated plan, with the real historical data from SCOOT sensors, we have observed that the PDDL+ model can simulate in an accurate way the evolution of the traffic network conditions for up to 15-30 minutes: the longer the period, the higher the probability of significant differences.

Validation of Generated Plans. Even though it is not shown in Figure 3, validation is a crucial step of the knowledge acquisition process, and of the deployment of the planning system. There exists a range of approaches to validate PDDL+ plans: VAL (Howey et al., 2004) is a well-known tool; planning engines such as ENHSP (Scala et al., 2020) include validation modules, and KE tools such as VS-studio can support validation by using VAL and providing visual representation of the plan. Existing validation tools are designed to return binary output about the validity of the plan with regards to the considered models, and maybe some additional information about PDDL+ events and processes that are not explicitly mentioned in plans. There is a lack of support for the identification of the reasons why a plan fails the validation check, and the suggestion of corrective actions. In PDDL+, the need is exacerbated by the fact that events and processes are automatically triggered and executed, and not shown (or not easy to follow) in the solution plan.

Dynamic-unpredictable Data. This kind of data is extremely hard to acquire, as it is heavily dependent on current circumstances and the way in which they affected the network. For instance, a road traffic accident on a link will reduce the capacity of the link, or some unplanned roadworks can change the valid traffic movements of a junction. From a modelling perspective, such cases can be handled in two ways, (i) by modifying the static data to reflect the changes in the topology and structure of the network, and (ii) by modifying the dynamic data appropriately. As an example of type (ii), the fact that a traffic movement is not allowed in a junction can be modelled by assigning it a value of 0.0 PCUs per second – i.e. no vehicles move through that. When possible, we are currently following the second approach, as it does not require to modify the static data and the corresponding pipeline. However, these changes have to be done manually with the support of a domain expert. Further, there are cases where this approach will not work, for instance in the extreme case of the complete failure of traffic lights within a junction. Such cases have to be manually addressed, to make sure that the knowledge encoded in the problem instance accurately reflects the modifications.

Uncertainty. Planning in the urban traffic control domain involves a significant degree of uncertainty. First, as described in the above paragraph, there are aspects that are basically impossible to predict. Second, as dynamic data is collected from sensors, there can be measurement errors, and sensors can be faulty. Measurement errors are quite common in the presence of SCOOT pressure sensors that cover multiple lanes: they tend to underestimate the volume of traffic as multiple vehicles crossing the sensors concurrently over different lanes are counted as a single vehicle. Third, the SCOOT system does provide sensors readings for a link only when the green time for that link terminates. In other words, there can be variable distances between two subsequent readings and, at the point in time when the initial state description of the planning problem instance is generated, some links will have more recent readings than others. This has the potential to increase the noise of the initial state. In our case study, the first class of uncertainty has been dealt with by the support of human experts that are manually describing how the event is affecting the model. The second is dealt with by including in the processing appropriate correction factors, and checking for unusual values that can indicate malfunctioning of a sensor. Finally, the third type of uncertainty is currently dealt with by averaging the values between two subsequent readings. In the future, we plan to employ an approach based on warm-ups, as used by other traffic simulators, where the planning system is run over a short period of time, to stabilise the modelled traffic conditions, before the actual planning process is started.

Transferability of the Acquisition Process. With regards to the knowledge acquisition process presented in the previous sections and shown in Figure 3, a question that naturally arises is: how easy would it be to transfer such process to a different urban region? In principle, the overall process that allowed to design the exploited knowledge acquisition architecture, that includes discriminating between static and dynamic data, identifying relevant data sources, etc., can be easily transferred between urban regions. However, the adaptors and parsers designed for the case study, as well as the designed approaches for validating and verifying the acquired knowledge, are not likely to be transferred easily if the current traffic control is other than SCOOT. On the other hand, the experience gained in the case study can be fruitfully exploited to speed up the process, and to avoid repeating mistakes. The above question can be stretched also to the transferability of the knowledge acquisition process to different application domains. The intuition is that the systematic approach that led to the design of the acquisition process can be transferred to different application domains, assuming the application domain does not involve life-critical operations. In that case, a significant effort has to be dedicated to ensuring that acquired knowledge is correct and safe. In the case of urban traffic control, this safety aspect is mitigated by the fact that traffic lights are forced to follow very strict regulations, and will ignore commands from the planning system that do not comply with such regulations.

6 CONCLUSION

In this paper we described the approach developed for generating on the fly complex problem instances, to be used for real-time planning of traffic light signals in a urban traffic network. Beside the need to generate instances on the fly, the complexity of the knowledge acquisition process is exacerbated by the different kind of data and multiple data sources involved – this is very different from the traditional way in which planning approaches are tested using simulators or thoroughly checked benchmark instances.

We exploited this opportunity to highlight the challenges that this kind of knowledge acquisition poses, and to present the solutions we used to address them in the case study. As a take-home message, we observed that there is a lack of support, in terms of tools and techniques, for the validation of generated solutions, the verification of the acquired knowledge, and the inspection of models. While this is partly due to the PDDL+ language, that does not allow to describe for instance the characteristics of valid states, there is also a lack of work in the area from the planning community, as highlighted by a recent analysis (Chrpa et al., 2017; Vallati and McCluskey, 2021).

We see several avenues for future work. First, we are interested in designing approaches to verify complex initial states expressed in PDDL+; this can be done either by leveraging on additional knowledge provided as an attachment to a planning model, or by analysing the characteristics of the domain model to identify suspicious trajectories. Second, we plan to extend the capabilities of existing validation approaches, to provide additional support when plans are analysed. Finally, we are working on a language for supporting the goal specification, that allows domain experts to express goals in a way that can then be translated into actual PDDL+ code.

ACKNOWLEDGEMENTS

Mauro Vallati was supported by a UKRI Future Leaders Fellowship [grant number MR/T041196/1].

REFERENCES

Barták, R., Fratini, S., and McCluskey, L. (2010). The third competition on knowledge engineering for planning and scheduling. *AI Mag.*, 31(1):95–98.

Capitanelli, A., Maratea, M., Mastrogiovanni, F., and Vallati, M. (2018). On the manipulation of articulated objects in human-robot cooperation scenarios. *Robotics and Autonomous Systems*, 109:139–155.

Cardellini, M., Maratea, M., Vallati, M., Boleto, G., and Oneto, L. (2021). In-station train dispatching: A PDDL+ planning approach. In *Proceedings of the Thirty-First International Conference on Automated Planning and Scheduling,*, pages 450–458. AAAI Press.

Chrpa, L., McCluskey, T. L., Vallati, M., and Vaquero, T. (2017). The fifth international competition on knowledge engineering for planning and scheduling: Summary and trends. *AI Mag.*, 38(1):104–106.

Fox, M. and Long, D. (2003). PDDL2.1: an extension to PDDL for expressing temporal planning domains. *Journal of Artificial Intelligence Research (JAIR)*, 20:61–124.

Fox, M. and Long, D. (2006). Modelling mixed discrete-continuous domains for planning. *J. Artif. Intell. Res.*, 27:235–297.

Fox, M., Long, D., and Magazzeni, D. (2012). Plan-based policies for efficient multiple battery load management. *J. Artif. Intell. Res.*, 44:335–382.

Ghallab, M., Nau, D., and Traverso, P. (2004). *Automated Planning: theory and practice*. Elsevier.

Gregory, P. (2020). PDDL Templating and Custom Reporting: Generating Problems and Processing Plans. In *Proc. of KEPS*.

Gulić, M., Olivares, R., and Borrajo, D. (2016). Using automated planning for traffic signals control. *PROMET-Traffic&Transportation*, 28(4):383–391.

Hoffmann, J. (2015). Simulated penetration testing: From "dijkstra" to "turing test++". In *Proceedings of the Twenty-Fifth International Conference on Automated Planning and Scheduling, ICAPS*, pages 364–372.

Hounsell, N. and McDonald, M. (1990). Astrid: Automatic scoot traffic information database. Project report.

Howey, R., Long, D., and Fox, M. (2004). VAL: automatic plan validation, continuous effects and mixed initiative planning using PDDL. In *Proceedings of the 16th IEEE International Conference on Tools with Artificial Intelligence (ICTAI)*, pages 294–301.

Hu, H. and Smith, S. F. (2019). Using bi-directional information exchange to improve decentralized schedule-driven traffic control. In *Proceedings of the Twenty-Ninth International Conference on Automated Planning and Scheduling, ICAPS*, pages 200–208. AAAI Press.

Kvarnström, J. and Doherty, P. (2010). Automated planning for collaborative uav systems. In *Control Automation Robotics & Vision (ICARCV), 2010 11th International Conference on*, pages 1078–1085.

Long, D., Dolejsi, J., and Fox, M. (2018). Building Support for PDDL as a Modelling Tool. In *Proc. of KEPS*.

Lopez, P. A., Behrisch, M., Bieker-Walz, L., Erdmann, J., Flötteröd, Y.-P., Hilbrich, R., Lücken, L., Rummel, J., Wagner, P., and Wießner, E. (2018). Microscopic traffic simulation using sumo. In *Proceedings of ITSC*.

McCluskey, T. L. and Porteous, J. M. (1997). Engineering and compiling planning domain models to promote validity and efficiency. *Artif. Intell.*, 95(1):1–65.

McCluskey, T. L. and Vallati, M. (2017). Embedding automated planning within urban traffic management operations. In *Proceedings of the Twenty-Seventh International Conference on Automated Planning and Scheduling, ICAPS*, pages 391–399.

McCluskey, T. L., Vaquero, T. S., and Vallati, M. (2017). Engineering knowledge for automated planning: Towards a notion of quality. In *Proceedings of the Knowledge Capture Conference, K-CAP*, pages 14:1–14:8.

Pozanco, A., Fernández, S., and Borrajo, D. (2021). Online modelling and planning for urban traffic control. *Expert Systems*.

Scala, E., Haslum, P., Thiébaux, S., and Ramírez, M. (2020). Subgoaling techniques for satisficing and optimal numeric planning. *J. Artif. Intell. Res.*, 68:691–752.

Smith, S. F. (2020). Smart infrastructure for future urban mobility. *AI Mag.*, 41(1):5–18.

Taale, H., Fransen, W., and Dibbits, J. (1998). The second assessment of the SCOOT system in Nijmegen. In *IEEE Road Transport Information and Control*, number 21-23.

Vallati, M. and Chrpa, L. (2019). On the robustness of domain-independent planning engines: The impact of poorly-engineered knowledge. In *Proceedings of the 10th International Conference on Knowledge Capture, K-CAP*, pages 197–204.

Vallati, M. and Kitchin, D. E., editors (2020). *Knowledge Engineering Tools and Techniques for AI Planning*. Springer.

Vallati, M., Magazzeni, D., De Schutter, B., Chrpa, L., and McCluskey, T. L. (2016). Efficient macroscopic urban traffic models for reducing congestion: a PDDL+ planning approach. In *The Thirtieth AAAI Conference on Artificial Intelligence (AAAI)*, pages 3188–3194.

Vallati, M. and McCluskey, T. L. (2021). A quality framework for automated planning knowledge models. In *Proceedings of the 13th International Conference on Agents and Artificial Intelligence, ICAART*, pages 635–644.

Xie, X.-F., Smith, S., and Barlow, G. (2012). Schedule-driven coordination for real-time traffic network control. In *Proceedings of the 22nd International Conference on Automated Planning and Scheduling (ICAPS)*, pages 323–331.