Self-Management in Urban Traffic Control – an Automated Planning Perspective

Falilat Jimoh and Thomas Leo McCluskey

School of Computing and Engineering
University of Huddersfield
{Falilat.Jimoh,t.l.mccluskey}@hud.ac.uk

Abstract. Advanced urban traffic control systems are often based on feed-back algorithms. They use road traffic data which has been gathered from a couple of minutes to several years. For instance, current traffic control systems often operate on the basis of adaptive green phases and flexible co-ordination in road (sub) networks based on measured traffic conditions. However, these approaches are still not very efficient during unforeseen situations such as road incidents when changes in traffic are requested in a short time interval. For such anomalies, we argue that systems are needed that can sense, interpret and deliberate with their actions and goals to be achieved, taking into consideration continuous changes in state, required service level and environmental constraints. The requirement of such systems is that they can plan and act effectively after such deliberation, so that behaviourally they appear self-aware.
This chapter focuses on the design of a generic architecture for autonomic urban traffic control, to enable the network to manage itself both in normal operation and in unexpected scenarios. The reasoning and self-management aspects are implemented using automated planning techniques inspired by both the symbolic artificial intelligence and traditional control engineering.Preliminary test results of the plan generation phase of the architecture are considered and evaluated.

Keywords: automated planning, urban traffic control, autonomic systems

1 Introduction

The need for planning and execution frameworks has increased interest in designing and developing system architectures which use state-of-the-art plan generation techniques, plan execution, monitoring and recovery in order to address complex tasks in real-world environments [1]. An example of such an architecture is T-Rex (Teleo-Reactive EXecutive): a goal oriented system architecture with embedded automated planning for on-board planning and execution for autonomous underwater vehicles to enhance ocean science [2, 3]. Another recent planning architecture, PELEA (Planning and Execution LEarning Architecture), is a flexible modular architecture that incorporates sensing, planning, executing, monitoring, replanning and even learning from past experiences [4].

This means that advanced control systems should be able to reason with their surrounding environment and take decisions with respect to their current situation and their desired service level. This could be achieved by embedding situational awareness into them, giving them the ability to generate necessary plans to solve problems themselves with little or no human intervention. We believe this is the key to embody autonomic properties in systems.

Self-managed systems (SM) are required to have an ability to learn process patterns from the past and adopt, discard or generate new plans to improve control systems. The ability to identify the task is the most important aspect of any SM element, this enables SM to select the appropriate action when healing, optimising, configuring or protecting itself.

Our self-managed system architecture is inspired by the functionality of the Human Autonomic Nervous System (HANS) that handles complexity and uncertainty with the aim to realise computing systems and applications capable of managing themselves with minimum human intervention.

2 Urban Traffic Control

There are several applications of artificial intelligence to urban traffic control with the aim of creating intelligent systems that will optimise the flow of traffic in urban centers [5–7]. Yet, the existing urban traffic control (UTC) approaches are not generally optimal during unforeseen situations such as road incidents, car breakdown or when traffic demand changes rapidly within a short time interval [8, 9]. This increases the need for autonomy in UTC. Such an autonomic system (AS) would need to be able to consider the factors affecting the situation at hand: the road network, the state of traffic flows, the road capacity limit, accessibility or availability of roads within the network etc. All these factors will be peculiar to the particular set of circumstances causing the problem [10]. Hence, there is a need for a system that can reason with the capabilities of the control assets, and the situation parameters as sensed by road sensors and generate a set of actions or decisions that can be taken to alleviate the situation. Therefore, we need systems that can plan and act effectively in order to restore an unexpected road traffic situation into a normal order. A significant step towards this is exploiting Automated Planning techniques which can reason about unforeseen situations in the road network and come up with plans (sequences of actions) achieving a desired traffic situation.

2.1 Role of AI Planning in Urban Traffic Control

The field of *Artificial Intelligence Planning*, or AI Planning, has evidenced a significant advancement in planning techniques, which has led to development of efficient planning systems [11–14] that can input declarative representation of models of applications. The existence of these general planning tools has motivated engineers in designing and developing complex application models which closely approximate real world problems. Thus, it is now possible to deploy

deliberative reasoning to real-time control applications [15, 16]. Consequently, AI Planning now has a growing role in realisation of autonomy in control systems and this architecture is a leap towards the realisation of such goal. The main difference between traditional and our autonomic control architecture is depicted in Figure 1. Traditionally, a control loop consists of three steps: sense, interpret and act [17]. In other words, data are gathered from the environment with the use of sensors, the system interprets information from these sensors as the state of the environment. The system acts by taking necessary actions which is feedback into the system in other to keep the environment in desirable state. Introducing deliberation in the control loop allows the system to reason and generate effective plans in order to achieve desirable goals. Enabling deliberative reasoning in UTC systems is important because of its ability to handle unforeseen situations which has not been previously learnt nor hard-coded into a UTC. Ultimately, this would help to reduce traffic congestion and carbon emissions.

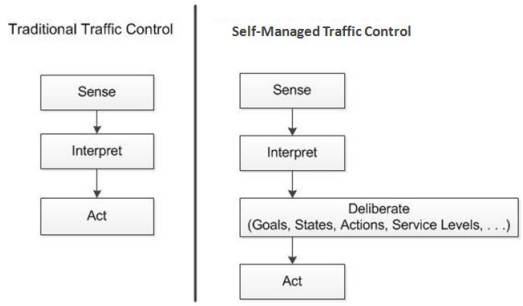

Fig. 1. Comparison of traditional and deliberative controls in Urban Transport Systems.

3 Automated Planning

AI Planning deals with the problem of finding a totally or partially ordered sequence of actions whose execution leads from a particular initial state to a state in which a goal condition is satisfied [18]. Actions in plans are ordered in such a way that executability of every action is guaranteed. Hence, an agent is able to transform the environment from an initial state into a desired goal state [19]. A planning problem thus involves deciding "what" actions to do, and "when" to do them [20].

In general, state space of AI planning tasks can be defined as a state-transition system specified by 4-tuple (S, A, E, γ) where:

- S is a set of states
- A is a set of actions

– $\gamma : S \times A \rightarrow 2^S$ is a transition function.

Actions can modify the environment (a state is changed after an action is triggered). Application of an action a in some state s results in a state in $\gamma(s, a)$ ($\gamma(s, a)$ contains a set of states). It refers to non-deterministic effects of actions.

Classical planning is the simplest form of AI planning in which the transition function γ is deterministic (i.e. $\gamma : S \times A \rightarrow S$). Hence, the environment is assumed to be static and fully observable. Also, in classical planning actions have instantaneous effects.

Temporal planning extends classical planning by considering time. Actions have durative effects which means that executing an action takes some time and effects of the action are not instantaneous. Temporal planning, in fact, combines classical planning and scheduling.

Conformant planning considers partially observable environments and actions with non-deterministic effects. Therefore, the transition function γ is non-deterministic (i.e. $\gamma : S \times A \rightarrow 2^S$).

For describing classical and temporal planning domain and problem models, we can use PDDL [21], a language which is supported by the most of planning engines. For describing conformant planning domain and problem models, we can use PPDDL [22] (an extension of PDDL) or RDDL [23], languages which are supported by many conformant planning engines. Our architecture can generally support all the above kinds of planning.

4 Autonomic Computing Paradigm

We use the term "autonomic system"(AS) rather than autonomic computing to emphasise the idea that we are dealing with a heterogeneous system containing hardware and software. Sensors and effectors are the main component of this type of autonomic system architecture [24]. AS needs sensors to sense the environment and executes actions through effectors. In most cases, a control loop is created: the system processes information retrieved from the sensors in order to be aware of its effect and its environment; it takes necessary decisions using its existing knowledge from its domain, generates effective plans and executes those plans using effectors. Autonomic systems typically execute a cycle of monitoring, analysing, planning and execution [25].

System architecture elements are self-managed by monitoring, behaviour analysing and the response is used to plan and execute actions that move or keep the system in a desired state. Overall self-management of a system is about doing self-assessment, protection, healing, optimisation, maintenance and other overlapping terms [24]. It is important to stress that these properties are interwoven. The existence of one might require the existence of the others to effectively operate. For instance, a self-optimisation process might not be complete until the system is able to self-configure itself. Likewise, an aspect of a proactive self-healing is an ability of the system to self-protect itself. A system that is meant to satisfy the above objectives will need to have the following attributes:

- Self-awareness of both internal and external features, process resources, and constraints
- Self-monitoring of its existing state and processes
- Self-adjustment and control of itself to the desirable/required state
- Heterogeneity across numerous hardware and software architectures.

In a quest to embed these autonomic attributes into an urban traffic control system, we explore the feasibility of embedding a automated planning component into an urban traffic control system. The next section gives an overview of the architectural design of our approach.

5 A Self-Managing System Architecture using Automated Planning

We describe the system's modules in the section below, using the diagram in Figure 2.

Key to the contents of the architecture is a declarative description of the system itself: the individual components that make up such a system; the dynamic environment that can influence the performance of such system; and its sensing and controlling capabilities. A novel feature is its ability to reason and deliberate using a combination of model predictive control (MPC) approach [26] and AI planning paradigm.

5.1 Road Traffic Network

A real traffic network contains information about the state of a road traffic environment at any time of the day.

Where historic data is available, or where behaviour is caused by anticipated circumstances, a road network will continue to flow normally using its programmed or adaptive intelligence. This means that signal heads will be able to control traffic as long as the traffic pattern is closed to the programmed or learned pattern [27]. Changes to the state of a traffic network are under the influence of various disturbances. For example, a sudden increase in volume of traffic because of the end of a football game in a nearby arena becomes a disturbance to a road network, if the network has not been previously programmed or adapted to such a traffic change.

Thus, our perception of the traffic situation within a network of roads is that a network of traffic will continue to operate as planned until a disturbance makes it act otherwise. In this work, a network of roads, as shown in Figure 3, within Huddersfield town center area of United Kingdom, was considered for our experimental analysis [28, 29].

5.2 Disturbances

These are the factors that affect the normal functioning of a road network. This includes planned and unplanned disturbances. An example of planned disturbance is a set of road works. This disrupts the normal flow of traffic for the

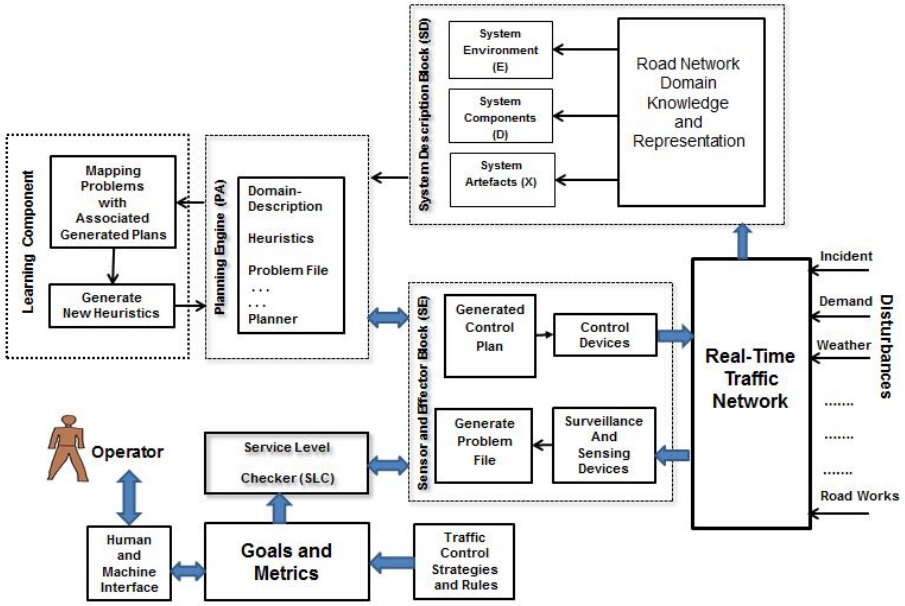

Fig. 2. Diagram of a Self-Managing System Architecture using Automated Planning

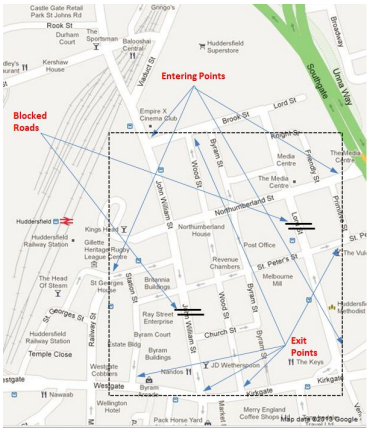

Fig. 3. Map showing the network entry and exit points and the blocked roads in a part of town centre of Huddersfield, West Yorkshire, United Kingdom. It is used for our empirical analysis.

entire duration of its construction. During such road works, the adjoining route and intersection signal need to be optimised in other to minimise the effect of road or lane blockage during the period of the maintenance or upgrade. There are several examples of unplanned disturbances. This includes but is not limited to road incident; unexpected change in traffic demand and severe weather conditions. Whenever a road incident occurs, the portion of the road would be blocked to allow comprehensive investigation for a period of time. During this period, the signal head should be aware of the status of the vehicles on the road through the use of road sensors and existing surveillance. Necessary alerts should be sent to the operator for medical emergency and security staff. Also the signal heads in adjoining routes should also be optimised to allow outflow of vehicles from the incident point to connected roads and the inflow of vehicle to the effected roads. This would be achieved automatically by a self-managed system by the use of message boards and switching all signal heads appropriately.

5.3 System Description Block (SD)

To create a self-aware property in a system, the system needs both to be situationaly aware, *and* have a knowledge model of its components, its environment and the relationship between them. The latter is achieved by extracting operational knowledge of a road traffic domain and representing that knowledge in a language that can be understood by the system. In the implementation of this work, much of the domain knowledge is represented in PDDL [21]. PDDL gives a formal representation of all the entities and operation policy of a road network in a language that is understood by an automated planning function.

The system description block comprises of the model of the road network domain. It is the storage for the declarative description of all the system components, the system environments and their components. It also gives a declarative description of the relationship between those components.

Formally, the system description block is a triple $\langle D, X, S \rangle$ where

- D is a finite set of system components
- X is a finite set of artefacts
- S is a set of states, where each state $s_i \in S$ is a set of assignments $f_k(y_l) = v_m$ where $y_l \in D \cup X$ and v_m is a value of y_l.

The components are all the objects that can be controlled by the system, for example, traffic control signal heads, variable message signs(VMS) and traffic cameras.

In the UTC domain, for instance, SD consists of a declarative description of the road network of an area that needs to be controlled while optimising the traffic flow patterns. This includes the representation of the road map as, for example, a directed graphs. Below is a snippet of the the connections between individual roads in our regional example:

```
. . .
(:objects
    markStr //there exist a road in the network called mark street
....

  ((:init
  (:init
   (conn JWInts north brookStr)//north of Brook street is connected
                                to south of John William's intersection.
   (link lordStr south brookStr) //south of Brook street is
                                   linked with Lord street
. . . .
```

The system components are the available objects in the system that can be manipulated and optimised for smooth traffic flow. This includes the signal heads and message boards. These objects have a set of properties and relations which record information, and changes to the objects are recorded through changes to these properties and relations. For example, an object called road has a name property called "byramStr" and a road capacity limit of 20 cars. Capacity limit is the maximum number of vehicle the road can contain at any giving period of time.

```
          ....
   (= (capacity brookStr) 11)//the maximum capacity of Brook street
                              is 20 cars
   (= (capacity byramStr) 20) //the maximum capacity for Byram
                              Street is 11 cars
          ...
```

This implies that whenever the road sensor shows a total of 20 vehicles on Byram street for a period of time without a dynamic change in the state of the vehicle(i.e not moving) - then such road is blocked and vehicles should be diverted from using it until the number of vehicles in such road is reduced.

5.4 Sensor and Effector Block (SE)

This is the sensory nerves of the system. It serves as the information interchange between the high level reasoning component of the system and the road network. It has an input mode and an output mode.

The Input Mode Surveillance and sensing devises are embedded into almost every road network. Road traffic monitoring agencies make use of surveillance and sensing devices for retrieving the status of road networks. These devices upload the real time data in a format that can be retrieved and understood by the system. The raw sensor data is then the start of an information processing pipeline which ends up as values that describe the properties and relationships of objects that are defined in the SD described in the section above.

For example, in the UTC domain, sensing can be done by road sensors and CCTV cameras. After information processing, this would result in a state file for urban traffic, for example:

```
. . .
(operational brookStr)// Brook street is operational with no
                         disruption
(operational byramStr)// Byram Street is operational with no
                         disruption
 . . .
(= (val north brookStr) 3)// number of vehicles on Brook Street
(= (val south byramStr) 12)// number of vehicles on Byram Street
  . . .
```

An excellent example of recovering information from an array of sensors, and extracting PDDL-like states and activities from them via information processing, is given in Laguna's recent thesis [30]. Here the sensors were RFID and camera, and the activity being sensed was cooking in a kitchen, but the parallel between this and the UTC domain is clear.

The Output Mode Control devices are called actuators in classical control terminology. They are embedded into systems to change their performance to a desirable state. Control devices play vital role in UTC environment, from the simple traffic light to the complex controller box for signal heads. They all help to control and maintain smooth traffic operation in road network. SE is responsible for executing control plans (sequences of actions) retrieved from the Planning and Action block. Generated control plans are received from this block, formatted and passed as system instructions to appropriate control devices for execution. The control devices change the state of the road network to desires state. An excerpt from the generated control file is:

```
. . .
25.010: (enable v2 north brookStr)  [50.000] //release vehicles on
        Mark Street for 50 seconds
26.007: (disAble v1 south JW)  [30.000] //stop vehicles on south of
        John Williams for 30 seconds
27.009: (divert-through-intersection y2 JWInts north brookStr)[60.000]
        //divert vehicles through south of john William's intersection
. . .
```

Executing actions is done by system components via the control devices. Since we have to consider uncertainty, after executing a sequence of actions, the sensors sense the current state, and if the current state is different from the expected one, the information is propagated to the Planning and Action block which provides a new plan.

5.5 Planning and Action (PA) Block

PA can be understood as a core of deliberative reasoning and decision making in the system. This is 'the brain' of our architecture. It has several components:

- an estimated current state
- a description of desired states or service levels[18]
- the model of the domain
- the planning problem file
- the planner program
- heuristics, such as ways of estimating how far a goal is away from a state.

The PA block receives an update to the estimated current state of the system from SE, and system goals (desired states or service levels input from the Service Level Checker, explained below). It produces a plan which aims to meet the system goals from the current state. The default planning technology to carry out this technique is AI Planning [18]. The plan produced is passed to the SE block for execution. However, it is well known from planning and execution systems [31] that producing a plan given a planning problem does not guarantee its successful execution. The SE block, which is responsible for plan execution, verifies whether executing an action provides the desired outcome. If the outcome is different (for whatever reason), then this outcome and current estimated state is passed back to the PA which is requested to re-plan.

In the UTC domain, if some road is congested, then the traffic is navigated through alternative routes. Hence, PA is responsible for producing a plan which may consist of showing information about such alternative routes on variable message signs and optimising signal heads towards such route. There are theoretical limits to the success of automated planning (even simple planning problems are intractable in general), however, and hence PA might not guarantee generating plans in a reasonable time. In a dynamic environment, where a system must be able to react quickly to avoid imminent danger, this may be a significant problem.

Real time processes require fast response times to deal with exogenous events. In order to cope with this we need to be able to satisfy two situations where:

- no valid plan can be found to meet the system goals
- the environment has changed prior to the generation and execution of a control plan.

These two planning problems can be minimised by utilising a Model Predictive Control(MPC) approach [26] to supplement an automated planner.

MPC techniques help to reduce the situation whereby the environment has changed prior to generation and execution of plans. Rather than using only the system model, the generating of the problem file takes all possible disturbances into consideration. It also sends the planning problem to the planner in a receding horizon approach. This means that the problem represents what is likely to happen in few seconds ahead. This allows the planner to generate plans with

consideration for future timing, and reduces the possibility of a totally changed environment prior to plan execution.

The MPC approach also helps to reduce the likelihood of situations where no valid plan can be found for given system goals.

This is achieved by breaking search exploration in to horizons such that the best node at every horizon is explored further if planner time limit is not exceeded. This is because exploring more search space will delay output time for a real time system, and the output time might not be predictable since the time taken to generate valid plans depends on the complexity of the task. This complexity varies at different stages of the entire process life span. The planner searches, considering all the possible combinations of the input constraints over a period of time and/or node counts and returns a sequence of steps that will take the road network from its current state to a partial or goal state. We use the phrase partial or goal state, because the planner will return a plan from its current state irrespective of whether the goals are met or not. This will make sure a plan is available to follow at every instance of time. Thus, allowing the planner to search for a limited resource count and returning the partial or complete goal plan at every time stamp, means that the planner will always generate a plan that will take the system closer to the goal trajectory provided that the fixed searching time and the node count are well guided.

Specifically, after searching for a fix time and/or node count, the planner communicates the partial or completely generated control plan to the network control devices through the sensor and effectors block, aiming to effect a change in state from an initial state to a fully or partially desirable state. The SE takes new samples from the reaction of the network to the effect of these sequences of control action. The network's new state with the corresponding dynamic input parameters from the environment is fed back into the planner to generate another sequence of steps that will take the network traffic from its current state to another partial/goal state for another sampling time, and so on.

This generates a control loop, which when continued over a period of time, will generate a continuous curve like that of an MPC as shown in Figure 4. This process takes the road network from its initial state to a goal state and maintains that goal state as long as the process is still active. The ability of the system to take the current state of the dynamic changing input parameters into consideration during planning process increases the robustness to unexpected events in the environments or system itself. This feedback loop also compensate for the few deviations in the production of sub-optimal plans. In fact, the smooth series of state changes, can only be achieved if the heuristics for the planner are sound. This means that the planner must always produce a correct (but not necessarily optimal) partial plan in the time allotted for plan generation.

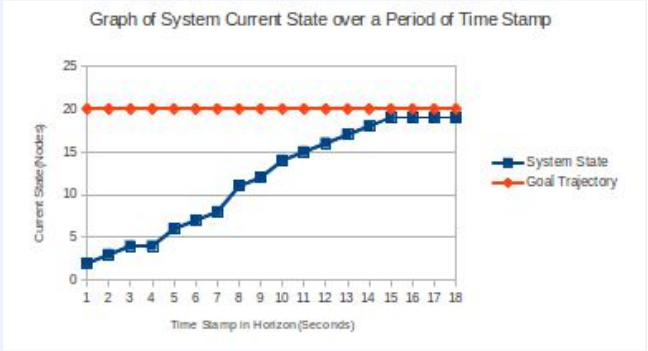

Fig. 4. Illustration of a well guided real-time process curve.

5.6 Learning Block

The inputs to the learning component are the generated plans and the associated problem files within the domain.

Generated plans and associated problems are mapped and stored for adaptation purposes.

The most common learning technique in UTC research systems is reinforcement learning: this learns appropriate actions for traffic patterns over a period of time, utilising the idea of assigning blame or positive feedback over repeated trials [27, 32]. Learning has also been implemented in several planning applications. The most common is the use of decision tree classifier to combine the generated plans with problem instances and learn a policy for the MDP(Markov decision process) from which the problem were drawn [33]. Thus, our emphasis in this part of the architecture is not in learning the pattern alone, but in generating new heuristics from the learning component to improve the performance of the planning engine. These heuristics includes macro actions for specific goals; the best horizon limit set for similar problems and the goal distance heuristics common to that environment at certain period of time. One clear advantage of the use of AI Planning in UTC is that the actions, goals and states are all defined in a declarative fashion. This means that why plans achieve a certain set of goals can be explained logically, using the operational semantics of the actions. Through this explanation we can derive the *weakest precondition* of a plan, in other words the smallest set of features on a state such that, if those features are seen again, the same plan can be re-used to achieve the same goals.

5.7 Service Level Checker (SLC)

The service level checker, as the name implies, repeatedly checks the service level of the entire system to see if the system is in a 'good condition'. It gets the current state of the system from SE and compares it with the 'ideal' service level. If the current state is far from being 'ideal', then the system should act in order to recover its state to a 'good condition'.

Prob.no.	Blockages	Crikey		Optic	
		runtime	makespan	runtime	makespan
1	No	0.16	156	0.19	53
1	Yes	0.14	156	0.16	64
2	No	1.22	122	0.35	43
2	Yes	1.28	170	0.40	52
3	No	36.84	299	1.79	57
3	Yes	51.52	446	1.54	70

Table 1. Our experimental results showing planners' performance in given settings. Runtime (in seconds) stands for a time needed by a planner to produce a plan. Makespan stands for a duration in which the plan can be executed.

Formally, we can define an error function $\epsilon : S \rightarrow \mathbb{R}_0^+$ which determines how far the current state of the system is from being 'ideal'. $\epsilon(s) = 0$ means that s is an 'ideal' state. If a value of the error function in the current state is greater than a given threshold, then SLC generates goals (desired states of a given components and/or artefacts) and passes them to the problem file to generate new goals and metrics.

An example of this in UTC domain can be seen in an accident scene at a junction. The 'ideal' service level for such a junction is to maximise traffic flow from the junction to neighbouring routes. The present state of the system shows that the traffic at that junction is now static with road sensors indicating static vehicles(i.e. vehicles are no longer moving). Such a situation is not 'ideal', the error function in such a state is high, hence, new goals are generated by SLC to re-route incoming traffic flowing towards such junction as well as divert existing traffic in such locations to a different route through connected roads.
It is also possible to alter the error function by an external system controller. For instance, a self-managed UTC system can also accept inputs from the traffic controller (Motoring Agencies). This means that the system behaviour can be altered by the user if needed.

6 Preliminary Evaluation

We have created a prototype architecture along the lines described in the sections above, integrating a UTC scenario, environmental knowledge etc with AI Planning engines, but in a simulated rather than real environment. The experimental setting consists of several 'specific' problems which are defined on the top of the road network as depicted in Figure 3.

Uncertainty is not considered in this experiment, due to the limitation of AI planning approaches to reason with uncertainty in a complex domain. Problem 1 addresses the problem of navigating five cars from Lord Street (bottom right corner of the map) to Wood Street (upper part of the map). Problem 2 addresses the problem navigating cars (two at each entry point) through the network such that they are evenly distributed at south, north and east exit points (individual vehicles are not distinguished). Problem 3 has the same settings but has five cars at each entry point. Problems 1-3 are considered in two different variants,

i.e., without blockages and with blockages. Experiments were run on Intel Core i7 2.9GHz, 5GB RAM, Ubuntu 12.04.1 LTS.

For evaluation purposes we selected known planning engines Crikey [12], Optic [34] and LPG-td [11] which are capable to handle PDDL 2.1 features (including timed initial literals). LPG-td, however, is not very useful for our problem by two reasons. Firstly, it does not successfully complete preprocessing when the problem has more than a few objects (e.g. roads). Secondly, it cannot handle well concurrency from actions sharing some fluent(s), i.e., actions having the same fluent in their descriptions are never considered as concurrent.

Table 1 shows the results of our preliminary experimental settings using the Crikey and Optic planners. We can see that Optic clearly outperforms Crikey in both criteria - makespan (a time needed for executing the plan) and runtime (a time needed by a planner to provide a plan). Moreover, Optic is an incremental planner, i.e., after finding a plan it searches for a better one (with lower makespan) until a given time limit is reached.

The results of Optic are very promising given the fact that plans have been retrieved in a very reasonable time (at worst slightly above one second) and their quality (makespan) is satisfactory. Cars can be therefore navigated reasonably through the road network even in case of some unexpected event which could lead to road blockage. Also, given the fact that in the real-world environment cars are entering the road network continuously, good quality plans (in terms of makespan) can somehow ensure that the traffic flow remains continuous and roads will not become congested.

A sample plan shows a solution of Problem 1 (no blockage) given by the Optic planner, shows some interesting aspects. Firstly, despite going from the same entry point to the same exit point cars are split along the way such that some cars are navigated through Northumberland Street (e1) and some cars are navigated through St. Peter Street (v2). This might be useful when some unexpected event occurs (e.g. an accident on St. Peter Street) and there is no time to redirect traffic. Secondly, the Optic planner is not very able to consider more than one car in a single action (nn1) even though it is possible. This is a shortcoming because it does not lead to optimal (or very nearly optimal) solutions. Basically, such a plan when executed allows one car to go through the junction, then the other cars have to wait until that car drives through the following road. This is quite counter-intuitive and consequently these plans are not optimal.

In summary, embedding planning components into the UTC might be beneficial since it enables (centralised) reasoning in the given area which, for instance, can more easily overcome non-standard situations (e.g. road blockage after an accident). Our results showed that the makespan of the plans (given by the Optic planner) did not increase much even though some roads were blocked. Minimising makespan, which is a goal of any UTC system, will help to reduce road congestion and pollution in the environment. However, as we demonstrated,using state-of-the-art domain-independent planning engines does not lead to optimal solutions even in quite simple cases. Also, it is questionable whether a plan-

ner's performance will not significantly decrease on larger road networks. On the other hand, developing a domain-dependent planner specifically tailored to RTDM might overcome (some of) these issues.

7 Challenges

This architecture is presently implementable in systems that have time delay. It cannot be implemented with real time processes that require continuous changes in nanoseconds/microseconds. This is due to the fact that prediction can be computationally demanding, so posing a challenge to implement it in real time. However with technology advancement on CPU processing speed and the introduction of High Performance Computing (HPC), future implementation are possible. It is also important to know that the controller might anticipate set-point changes which may not be desirable. This could be as a result of a grossly inaccurate model which will yield poor control decisions, although the method is surprisingly robust. To achieve a highly tuned controller, a very accurate model is needed, because one can only control as precisely as one can model. The sampling of the environment at every period of time(state selection) for control plan generation should also encompass all significant dynamics and disturbances, otherwise performance may be poor and important events may be unobserved.

The most important lesson learnt through our experiments with this deliberative architecture is that the heuristics that the planners use must be well guided. This is directing our research into developing specialised planners which are tuned for UTC use. This will allow us to take advantage of the peculiarities of the application area, and result in operationally successful planners.

8 Conclusion

This chapter describes our vision of self-management at the architectural level within a UTC management system, where autonomy is the ability of the system components to configure their interaction in order for the entire system to achieve a set service level. We created an autonomic architecture made of several blocks which are discussed in the context of the Urban Traffic Control Domain. We described the functionality of each block, highlighting their relationship with one another. We also highlighted the role of AI planning in enabling a self-management property in an autonomic systems architecture. We believe that, creating a generic architecture that enables control systems to automatically reason with knowledge of their environment and their controls, in order to generate plans and schedules to manage themselves, would be a significant step forward in the field of autonomic systems.

In a real-world scenario where data such as road queues are uploaded in real-time from road sensors with traffic signals connected to a planner, we believe that our approach can optimise and re-store road traffic disruption with little or no human intervention. To that end, the development of specialised, deliberative AI planners, tuned to UTC applications, is necessary.

References

1. Myers, K.L.: Towards a framework for continuous planning and execution. In: In Proceedings of the AAAI Fall Symposium on Distributed Continual Planning, AAAI Press (1998)
2. McGann, C., Py, F., Rajan, K., Thomas, H., Henthorn, R., McEwen, R.: A Deliberative Architecture for AUV Control. In: Intnl. Conf. on Robotics and Automation (ICRA), Pasadena (May 2008)
3. Pinto, J., Sousa, J., Py, F., Rajan, K.: Experiments with Deliberative Planning on Autonomous Underwater Vehicles. In: IROS Workshop on Robotics for Environmental Modeling, Algarve, Portugal (2012)
4. Jimnez, S., Fernndez, F., Borrajo, D.: Integrating planning, execution, and learning to improve plan execution. Computational Intelligence **29**(1) (2013) 1–36
5. Smith, S., Barlow, G., Xie, X.F., Rubinstein, Z.: Surtrac: Scalable urban traffic control. In: Transportation Research Board 92nd Annual Meeting Compendium of Papers, Transportation Research Board (January 2013)
6. Tettamanti, T., Luspay, T., Kulcsar, B., Peni, T., Varga, I.: Robust control for urban road traffic networks. IEEE Transactions on Intelligent Transportation Systems **15**(1) (2014) 385–398
7. Mitsakis, E., Salanova, J.M., Giannopoulos, G.: Combined dynamic traffic assignment and urban traffic control models. Procedia - Social and Behavioral Sciences **20** (2011) 427–436
8. Roozemond, D.A.: Using intelligent agents for pro-active, real-time urban intersection control. European Journal of Operational Research **131**(2) (2001) 293–301
9. De Oliveira, D., Bazzan, A.L.C. In: Multiagent Learning on Traffic Lights Control: Effects of Using Shared Information. (2009) 307–322
10. Jimoh, F., Chrpa, L., Gregory, P., McCluskey, T.: Enabling autonomic properties in road transport system. In: The 30th Workshop of the UK Planning And Scheduling Special Interest Group, PlanSIG 2012. (2012)
11. Gerevini, A., Saetti, A., Serina, I.: An approach to temporal planning and scheduling in domains with predictable exogenous events. Journal of Artificial Intelligence Research (JAIR) **25** (2006) 187–231
12. Coles, A.I., Fox, M., Long, D., Smith, A.J.: Planning with problems requiring temporal coordination. In: Proc. 23rd AAAI Conf. on Artificial Intelligence. (July 2008)
13. Penna, G.D., Intrigila, B., Magazzeni, D., Mercorio, F.: Upmurphi: a tool for universal planning on pddl+ problems. In: Proc. 19th Int. Conf. on Automated Planning and Scheduling (ICAPS). (2009) 19–23
14. Eyerich, P., Mattmüller, R., Röger, G.: Using the Context-enhanced Additive Heuristic for Temporal and Numeric Planning. In: Proc. 19th Int. Conf. on Automated Planning and Scheduling (ICAPS). (2009)
15. Lhr, J., Eyerich, P., Keller, T., Nebel, B.: A planning based framework for controlling hybrid systems. In: International Conference on Automated Planning and Scheduling. (2012)
16. Ferber, D.F.: On modeling the tactical planning of oil pipeline networks. In: Proc. 18th Int. Conf. on Automated Planning and Scheduling (ICAPS), AAAI (2012)
17. Gopal, M.: Control systems: principles and design. McGraw-Hill, London (2008)
18. Nau, D., Ghallab, M., Traverso, P.: Automated Planning: Theory & Practice. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA (2004)

19. Garrido, A., Onaindia, E., Barber, F.: A temporal planning system for time-optimal planning. In: Progress in AI. Volume 2258 of LNCS. (2001)
20. Hoffmann, J., Nebel, B.: The FF planning system: Fast plan generation through heuristic search. Journal of Artificial Intelligence Research (JAIR) **14** (2001) 253–302
21. McDermott D. et al.: PDDL–the planning domain definition language. Technical report, Available at: `www.cs.yale.edu/homes/dvm` (1998)
22. Younes, H.L.S., Littman, M.L., Weissman, D., Asmuth, J.: The First Probabilistic Track of the International Planning Competition. J. Art. Int. Res. (JAIR) **24** (2005) 851–887
23. Sanner, S.: Relational dynamic influence diagram language (RDDL): Language description. http://users.cecs.anu.edu.au/ ssanner/IPPC_2011/RDDL.pdf (2010)
24. Ganek, A., Corbi, T.: The dawning of the autonomic computing era. IBM Systems Journal **42**(1) 5–5 00188670 Copyright International Business Machines Corporation 2003 2003.
25. Lightstone, S.: Seven software engineering principles for autonomic computing development. ISSE **3**(1) (2007) 71–74
26. Camacho, E., Bordons, C.: Model predictive control. Springer, London (1999)
27. Bazzan, A.L.: A distributed approach for coordination of traffic signal agents. Autonomous Agents and Multi-Agent Systems **10**(1) (January 2005) 131–164
28. Shah, M.M.S., Chrpa, L., Kitchin, D.E., McCluskey, T., Vallati, M.: Exploring knowledge engineering strategies in designing and modelling a road traffic accident management domain. In: 23rd International Joint Conference on Artificial Intelligence (IJCAI 2013), AAAI (2013)
29. Jimoh, F., Chrpa, L., McCluskey, T.L., Shah, S.: Towards application of automated planning in urban traffic control. In: Intelligent Transportation Systems-(ITSC), 2013 16th International IEEE Conference on, IEEE (2013) 985–990
30. Laguna, J.O., Olaya, A.G., Millan, D.B.: Building Planning Action Models Using Activity Recognition. PhD thesis, Universidad Carlos III de Madrid (2014)
31. Fox, M., Long, D., Magazzeni, D.: Plan-based policy-learning for autonomous feature tracking. In: International Conference on Automated Planning and Scheduling. (2012)
32. Dusparic, I., Cahill, V.: Autonomic multi-policy optimization in pervasive systems: Overview and evaluation. ACM Trans. Auton. Adapt. Syst. **7**(1) (May 2012)
33. Fox, M., Long, D., Magazzeni, D.: Plan-based policies for efficient multiple battery load management. Computing Research Repository **abs/1401.5859** (2014)
34. Benton, J., Coles, A.J., Coles, A.: Temporal planning with preferences and time-dependent continuous costs. In: ICAPS. (2012)