A Synthesis of Automated Planning and Model Predictive Control Techniques and its Use in Solving Urban Traffic Control Problem

Falilat Jimoh

School of Computing and Engineering

University of Huddersfield

A thesis submitted for the degree of

Doctor of Philosophy

2015

I would like to dedicate this thesis to my loving husband, Abdul Wasiu Adetunji Jimoh, for his continuous trust and support during my endless challenges of life.

Acknowledgements

All praises are due to Almighty Allah for making it possible for me to complete my Ph.D. programme. I give thanks to Allah for sparing the life of my parents to witness this wonderful moment of their child. I pray Allah grant them longer life in good health and faith. I also give thanks to Almighty Allah for given me the best of spouse and healthy children. All praises are due to Allah for making it possible for my children to endure pains of living without their mother. May we all live long to appreciate the blessing of Allah upon our family.

I would like to acknowledge the contribution of my siblings and in-laws for making it possible for me to stay abroad while they take care of my love ones. My special appreciation goes to "big mummy" for her priceless sacrifices to take care of our old parent and still keep my family together while being away. My special appreciation also goes to two jewels among my in-laws: "Ms Peju Udoinwang" and "Engr Bashir Jimoh" for their priceless sacrifice and effort during the completion stage of this Ph.D. programme despite my weak and fragile state of health.

I would like to acknowledge the help of my faculty members in my academic achievement, most especially my supervisor Professor T. L. McCluskey for his fatherly support and advice during most of my health and family challenges. My special appreciation also goes to Dr Lukas Chrpha for his advice and support during this Ph.D. programme especially in his role in most of my academic publications.

My acknowledgement also goes to the research office staffs for making my Ph.D. life seamless despite all the frictions in my research profiles and documentation. My special appreciation goes to Ms Gwen Wood and Mr Chris Sentence for going out of their way to put a smile on the face of students like me during difficult times.

I would like to acknowledge the prayers, advice and support of most of our family friends: The AbdulRaman's for always being by our side from the conception of this idea till the actualisation of the research. The Hassan's the Abdul-Kareem's for their continuous encouragements. The Akanni's for their priceless duas. The Qahaar's for their physical, emotional and spiritual support towards making me feel comfortable home away from home. The Naheed's for their support and always being there at my time of need. The Hamzat's for their social upliftment in the time of depression away from home. My appreciation also goes to the rest of my family friends that have believed and contributed in this course of mine. The list is endless, but Allah knows you all. May Almighty Allah in His infinite mercy grant you the highest place in paradise.

My acknowledgement also goes to my colleagues in the office for their sincere advice and guidance during the difficult time of my Ph.D. programme. A special acknowledgement goes to Dr Aisha Jilani and Dr Shahin Shah for the guidance and supports they gave to me especially during the early years of my Ph.D.

My final acknowledgement goes to most of my friends for their prayers and advice before and during this Ph.D. programme. Listing their names could be as long as this thesis chapters. Almighty Allah knows you all and would reward you with the best of reward in this world and beyond.

Abstract

Most desired applications for planning and scheduling typically have the characteristics of a continuous changing world. Unfortunately, traditional classical planning does not possess this characteristic. This drawback is because most real-world situations involve quantities and numeric values, which cannot be adequately represented in classical planning. Continuous planning in domains that are represented with rich notations is still a great challenge for AI. For instance, changes occurring due to fuel consumption, continuous movement, or environmental conditions may not be adequately modelled through instantaneous or even durative actions; rather these require modelling as continuously changing processes. The development of planning tools that can reason with domains involving continuous and complex numeric fluents would facilitate the integration of automated planning in the design and development of complex application models to solve real world problems.

Traditional urban traffic control (UTC) 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 need systems that can plan and act effectively in order to restore an unexpected road traffic situation into a normal order. In the quest to improve reasoning with continuous process within the UTC domain, we investigate the role of Model Predictive Control (MPC) approach to planning in the presence of mixed discrete and continuous state variables within a UTC problem. We

explore this control approach and show how it can be embedded into existing, modern AI Planning technology. This approach preserves the many advantages of the AI Planning approach, to do with domain independence through declarative modelling, and explicit reasoning while leveraging the capability of MPC to deal with continuous processes.

We evaluate the possibility of reasoning with the knowledge of UTC structures to optimise traffic flow in situations where a given road within a network of roads becomes unavailable due to unexpected situations such as road accidents. We specify how to augment the standard AI planning engine with the incorporation of MPC techniques into the central reasoning process of a continuous domain. This approach effectively utilises the strengths of search-based and model-simulation-based methods.

We create a representation that can be used to capture declaratively, the definitions of processes, actions, events, resources resumption and the structure of the environment in a UTC scenario. This representation is founded on world states modelled by mixed discrete and continuous state variables. We create a planner with a hybrid algorithm, called UTCPLAN that combines both AI planning and MPC approach to reason with traffic network and control traffic signal at junctions within the network. The experimental objective of minimising the number of vehicles in a queue is implemented to validate the applicability and effectiveness of the algorithm. We present an experimental evaluation showing that our approach can provide UTC plans in a reasonable time. The result also shows that the UTCPLAN approach can perform well in dealing with heavy traffic congestion problems, which might result from heavy traffic flow during rush hours.

Contents

List of Figures

Chapter 1

Introduction

1.1 Motivation

The field of *Artificial Intelligence Planning*, or AI Planning, deals with the problem of finding a sequence or partially ordered set of actions whose execution leads from an initial state to a state in which a goal condition is satisfied. Actions in plans are ordered in such a way that executability of every action is guaranteed. Hence, it is now possible to transform the environment from an initial state into a desired state that satisfies some goal conditions. A planning problem thus involves deciding "what" actions to do, and "when" to do them. The "when" part of the problem refers to "scheduling".

Most desired application for planning and scheduling typically have the following characteristics: a continuous changing world; multiple agents (both cooperative and adversarial); uncertainty of the state of the world; communication with external sources (sensors, databases, human users) to get information about the world; overlapping actions; time constraints and numeric computations involving resources, probabilities, spatial relationships and geometric. Unfortunately, traditional classical planning does not possess any of these characteristics.

This drawback of classical planning is due to the fact that most real-world situations involve quantities and numeric values, which are not properly represented in classical

planning. The need to express numeric quantities of entities within a domain representation led to the development of metric planning [42; 73]. Also, continuous planning in domains that are represented with rich notations has long been a great challenge for AI [35]. For instance, changes occurring due to fuel consumption, continuous movement, or environmental conditions may not be adequately modelled through instantaneous or even durative actions; rather these require modelling as continuously changing processes. The combination of time-dependent problems and numeric optimisation problem create a more challenging and hard task of time-dependent metric fluents.

Thus, automated planning had to move away from the restricted classical planning with the introduction of real world problems which included some non-classical planning. From 2002, there has been an introduction of elementary notation of timing and resources. In 2004, inference rules and derived effects were added with a new track of planning in probabilistic domains. 2006 witness addition of soft goals, trajectory constraints, preferences, plan metrics, and expressions of constraints in temporal logic. AI Planning has evidenced a significant advancement in planning techniques in the last 10 - 15 years, which has led to the development of efficient planning systems that can input expressive models of applications. The existence of these general planning tools has motivated engineers in designing and developing complex application models that closely approximate real-world problems. These improvements enable automated planning application to cut across several disciplines with different applications ranging from space exploration, education, health care, business and commerce to entertainment industry.

Urban traffic management is an application area that we believe could benefit from the applications of AI Planning and is the motivating application for this thesis. Current urban traffic management centres (UTMCs) have systems that help to minimise delay within day to day traffic flows using controls such as self-tuning traffic light clusters. They are effective in coping with local expected changes in traffic flows, but are **not** designed, however, to adjust flow **regionally**. Demands such as re-routing vehicles to

avoid poor air quality hotspots, utilising the network fully, or working adequately in the face of unplanned exceptional events such as road closures or accidents, cannot be met by Urban Traffic Management Centres on a day to day basis. For instance, UTC operators have no technological help to synthesise and coordinate collections of interventions (light phase changes, message signs) to influence the build-up of traffic related flow over a region. Throughout the world urban operators, and hence road users, would benefit from tools to support the effective regional management of roads. Integrating the knowledge of control engineering into the field of AI planning and scheduling to create hybrid technology that can solve these kinds of complex planning problems of regional management could prove the foundation for a breakthrough for the transportation industry.

The area of AI Planning and Scheduling (AIP & S) involves modelling a system in a knowledge-based way, with declarative data structures representing goals, states, resources, actions to mention a few, and creating tools that can reason about them logically. Plans are created as output to achieve goal conditions in a future state. In this case, goals and actions can be easily changed or even generated themselves automatically (for example, a trigger on pollution monitoring data feeds could lead to the generation of goals to be fed into a planning engine that would output the appropriate plans to be carried out by traffic controls). As well as the flexibility of input language, a characteristic of this approach is that the human user can understand and inspect the output plan, to help validate the approach, and to promote a mixed-initiative interaction with the system operators. There are drawbacks, however, in that few if any planning engines can reason with models described by continuous processes, and the computational complexity of the planning problem can be prohibitive without strong heuristics.

This thesis addresses these drawbacks and proposes a planning algorithm that can reason with continuous processes using approach that are aimed at handling the computational complexity of the UTC planning problem.

Control Engineering, on the other hand, have developed techniques to control a continuous process by the varying of a control variable over a fixed time horizon. The motivation for this originates in areas of chemical engineering. Here the process is modelled by some approximating function, and the ”goal” is to ensure that certain of the variables of the dynamic system stay within a specified interval. A simple example is to adjust gas on a burner to ensure a pan will simmer but not boil over. The advantages of Control Engineering approaches in this area, such as embedded in the popular “Model Predictive Control” technique, is that they can work with dynamical systems described as continuously changing processes, and output controls that take into account future events. On the other hand, these approaches are not as flexible, nor as transparent, as the AIP & S approaches.

This thesis aims to transform areas of control engineering and AIP & S by formulating plan generation algorithms, and execution architectures, which combine the advantages of both approaches into one unified approach. The algorithms can produce readable plans to achieve goal conditions where the dynamical system model includes continuous processes. This thesis will lead to the creation of shared, flexible, generic tools to perform planning capable of being used in UTC management. It will lead to a number of specifications of sample problems that can be used to characterise the application of the technology. The planner produced is inspired and motivated by urban traffic control problem: the output technology is tested as part of the solution to UTC control for UMTCs. Our present effort is focused on using a Model Predictive Control(MPC) architecture to implement a continuous planner that can generate plans that can serve as advisory to traffic officers as well as urban traffic controller(agent) in other order to optimize traffic flows in urban areas.

In this thesis, we explore the application of AI planning technology and control engineering to the problem of Urban Traffic Control(UTC) system. We design a generic architecture for self-management systems which is inspired by Human self-management Nervous System. Our architecture consists of several components, which is discussed

in the context of the Urban Traffic Control Domain. We investigate the feasibility of using this approach in an urban road traffic domain to generate plans, where learned or reactive behaviours fail because of unforeseen situations. We show how this is accomplished by the creation of a declarative representation of a road network in a town centre area in the United Kingdom. The task is to navigate effectively cars through a network of connected roads during an unforeseen situation.

1.2 Thesis Contribution

This thesis describes a continuing effort to unite control engineering technique with automated planning approach towards solving controls problems in the area of urban traffic management. Our contributions are as follows, the first being the most important.

- The design, implementation and testing of a planner algorithm which can input expressive domain descriptions, output solution plans containing continuous processes, events and actions. This work is the first AI Planner we are aware of to integrate MPC with AI search - based planning techniques.
- An investigation into the role of Model Predictive Control(MPC) architecture in automated planning.
- The Design of a hybrid planner algorithm that is angled towards applications exemplified by the urban road traffic management problem
- Modelling of a declarative representation of the road network of Huddersfield town centre area of West Yorkshire in the United Kingdom. Thus, creating another benchmark domain for the planning community
- The design of a generic UTC architecture for self-management systems which is inspired by Human self-management Nervous System.

- Evaluating the possibility of reasoning with knowledge from UTC domain and optimising traffic flow in situations where a given road within a network of roads becomes unavailable due to an unexpected situation such as a road accident
- Evaluating the advantages and challenges of using AI Planning Technology in Urban Traffic Control

1.3 Thesis Layout

This thesis is divided into eight main chapters, the contents of individual chapters are summarised below.

Chapter One Chapter one explains the introduction to this research work. It discusses our motivation for this piece of work and the contribution it has to knowledge. It also gives a brief outline of the content in individual chapters of the entire thesis.

Chapter Two Chapter two explains the concept of Automated Planning as a branch of artificial intelligence and how it applies to real-world problems. It gives an overview of various terminologies in Automated Planning used by different chapters in this thesis. It also gives an insight into what constitutes an automated planning approach with a detailed explanation of planning language semantics and their associated planning engines called "planners". This chapter ends with an overview of some applications of automated planning in the different field of knowledge.

Chapter Three This chapter introduces Urban Traffic Control Problem.

Chapter Four This chapter explores the application of AI planning technology to the problem of Urban Traffic Control(UTC). This application is accomplished by the creation of a declarative representation of a road network in a town center area in the United Kingdom. The task is to navigate cars effectively through a

network of connected roads during an unforeseen situation. As part of this effort, we embed the knowledge of UTC structure into a planning domain and evaluate the possibility of reasoning with this knowledge and optimising traffic flow in situations where a given road within a network of roads becomes unavailable due to an unexpected situation such as road accidents. From this simulation, we demonstrate the advantages and shortcomings of using AI Planning in UTC.

Chapter Five This chapter introduces model predictive control as a branch of control engineering. It explains the strategy and approach of MPC in controlling and stabilising process control within the field of Engineering. This chapter ends by comparing the similarities and difference between MPC and AI planning to make a judgement call for a hybrid solution that incorporate both technologies to solve problems involving both discrete and continuous state variables.

Chapter six This chapter introduces Urban Traffic Control and the design of a generic architecture that is aimed at using both AI planning and MPC approach to creating self-management properties in UTC. In this chapter, the characteristics of a self-managed system and the need for a certain level of self-management in UTC is discussed. It explains the design of a generic self-management architecture which is inspired by Human self-management Nervous System. Our architecture consists of several blocks. This chapter describes each of the blocks in the context of the Urban Traffic Control Domain.

Chapter Seven This chapter describes our efforts towards using a Model Predictive Control (MPC) approach to creating a hybrid planner. It explains an urban traffic control planning (UTCPLAN) algorithm and implementation, which is aimed at, integrating a control engineering techniques with automated planning approach to create a hybrid planning system that can reason with domains containing discrete and continuous state variables such as found in UTC problems.

Chapter Eight This is an evaluation chapter that discusses and analyse the result of our experiments. It also discusses the challenges in this research work.

Chapter Nine This is the concluding chapter of this thesis. It summarises the overall work that has been accomplished so far and gives the future direction of the next step in our implementation process.

1.4 Publications

This section presents the list of publications that were achieved during the Ph.D. research in order to support and validate the scientific quality of this thesis.

1. Jimoh, F. and McCluskey, T.L. (2012), *Using Automated Planning to Enable self-management Properties in Computer Systems*, In the Proceedings of The Queens Diamond Jubilee Computing and Engineering Annual Researchers Conference 2012: CEARC12. University of Huddersfield, Huddersfield, p. 150. ISBN 978-1-86218-106-9

2. M. M. S. Shah, L. Chrpa, F. Jimoh, D. Kitchin, T. L. McCluskey, S. Parkinson and M. Vallati (2013), *Knowledge Engineering Tools in Planning: State-of-the-art and Future Challenges*, In the Proceedings of ICAPS Workshop on Knowledge Engineering for Planning and Scheduling (KEPS), June 10-14, Roam, Italy.

3. F. O. Jimoh, L. Chrpa, T. L. McCluskey and M. M. S. Shah (2013), *Towards Application of Automated Planning in Urban Traffic Control*, In the Proceedings of 16^{th} International IEEE Annual Conference on Intelligent Transportation Systems, October 6-9, The Hague, The Netherlands.

4. M. M. S. Shah, T. McCluskey, P. Gregory and F. Jimoh (2012), *Modelling Road Traffic Incident Management Problem for Automated Planning*, 13^{th} IFAC Sym-

posium on Control in Transportation System (CTS12), September, 12-14. Sofia, Bulgaria.

5. F. O. Jimoh, TL McCluskey, L. Chrpa (2012) and Peter Gregory *Application of Automated Planning to self-management Road Transport Support System*. In the proceedings of 30th Workshop of the UK Planning And Scheduling Special Interest Group (PlanSig12), University of Teesside, Middlesbrough, UK.

6. M. M. S. Shah, L. Chrpa, P. Gregory, T. L. McCluskey and F. Jimoh (2012), *OCLplus: Processes and Events in Object-Centred Planning*, the 6^{th} Starting Artificial Intelligence Research Symposium, August, 27-28. Montpellier, France.

7. F. O. Jimoh and Holmes, Violeta (2012), *Design of a wireless patient diagnosis system*, In the Proceedings of The Queens Diamond Jubilee Computing and Engineering Annual Researchers Conference 2012: CEARC12. University of Huddersfield, Huddersfield, pp. 69-74. ISBN 978-1-86218-106-9

8. I Cenamor, L Chrpa, F Jimoh, TL McCluskey, M Vallati (2014) *Planning and Scheduling Applications in Urban Traffic Management*In the Proceedings of 32nd Workshop of the UK Planning And Scheduling Special Interest Group (PlanSig12), University of Teesside, Middlesbrough, UK.

9. Jimoh, Falilat, Chrpa, Luk and Vallati, Mauro (2013), *self-management System Architecture: An Automated Planning Perspective*. In: Artificial Intelligence Applications and Innovations 2013. IFIP Advances in Information and Communication Technology, 412 (121). Springer, Pahpos, Cyprus, pp. 121-130. ISBN 9783642411410

10. Jimoh, Falilat and McCluskey, T.L. (2013), *Self-management in road traffic networks*. In: Proceedings of Computing and Engineering Annual Researchers' Conference 2013: CEARC'13. University of Huddersfield, Huddersfield, pp. 73-79. ISBN 9781862181212

11. Jimoh, Falilat, Chrpa, Lukas and McCluskey, T.L. (2014) The Application of Planning to Urban Traffic Control. In: SPARK Workshop of 24th International Conference on Automated Planning and Scheduling, 21st to 26th of June 2014, Portsmouth, NH, USA.

12. Jimoh, Falilat and McCluskey, T.L. (2015), *Self-Management in Urban Traffic Controlan Automated Planning Perspective* : Towards Autonomic Road Transport Support System. Birkhuser/Springer, 2015.

Chapter 2

Introduction to Automated Planning Approach

The ability to reason with the dynamics of life and its environment by creating and implementing plans to solve everyday challenges; is one of the uniqueness of human race. Embedding this quality of man into artificial entities such as machines, is the foundation of Automated Planning. This makes Automated planning a branch of Artificial Intelligence, which deals with the problem of finding a sequence or partially ordered set of actions whose execution leads from an initial state to a state in which a goal condition is satisfied. Actions in plans are ordered in such a way that executability of every action is guaranteed within the model [57]. A *Planning Problem* could consist of a set of concrete objects, an initial state, a goal condition and available actions that can be performed in each situation. An *initial state* is represented by a set of assignments and predicates that are true at the initial state. The predicates and assignment give the abstraction of the state of the world at any specific time. Fig 2.1 is an example of predicates and assignment from the tea_making environment.A planning problem thus involves deciding "what" actions to do, and "when" to do them. The "when" part of the problem refers to "scheduling" [63].

```
in(water, kettle) // means water is in the kettle
on(kettle) // means the kettle is switched on
. . .
```

Figure 2.1: Sample Predicates and Assignment in the Tea Making Environment

```
(action :switch_off
        :parameter [kettle]
        :preconditon on(kettle)//means kettle is ON
        :effect off(kettle) // means kettle is now OFF
)
```

Figure 2.2: Sample Action Declaration in the Tea Making Environment

A *goal situation* is could be represented by a set of relational, logical expressions or both (assignments and predicates) that are valid at the goal state. *Actions*, on the other hand, are the list of operators that changes the state of an environment from one state to another whenever it precondition is satisfied. Figure 2.2 is an example of action declaration in the tea_making environment is:

A plan is a *solution* of a given planning problem if and only if every action is applicable in the given time-stamp (its precondition is fulfilled) and after all action are executed all expressions specified in the goal situation are satisfied. For example,the plan to make a cup of tea might include the steps show in Figure 2.3.

The sample plan above shows; the *time* each line of statement should be executed;the actions that should be performed at every step; the objects of those actions and the time duration for the execution of each line of action.

```
Time  Action    Parameter              Duration
0:001 Get_Water [water,tap, kettle]    3:00
3:000 Switch_On [Water,kettle]         10:00
3:100 Get_TeaBag [teaBag,cup]          3:00
3:000 Pour_Water [water,cup]           2:00
. . .
```

Figure 2.3: Sample Plan to Make a Cup of Tea

Technically, 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
- E is a set of events
- $\gamma : S \times (A \cup E) \rightarrow 2^S$ is a transition function

A *Planning Domain*(description of an environment) consists of a set of predicates and numeric fluents describing the environment and a set of planning operators. One of the ways of modelling a *Planning Operator* is to specify via its precondition and effects. The *precondition* of a planning operator is represented by a set of relational or logical expressions or both. *Effects* are represented by a set of assignments or set of literals (predicates or their negations) or both. An *action* on the other hand, is a ground instance of a planning operator that can modify the environment. Application of an action a in some state s results in a state in $\gamma(s,a)$, where $\gamma(s,a)$ contains set of states.

2.1 Planners

The concept of a planner in the field of Automated planning is similar to a brain box of a system. It takes in the knowledge of the environment(domain) and the its associated problems(planning problem) to produce a sequence of solution that would change a given situation(initial state) to a desirable state(goal state). A planner's input comprises of the planning problem, with initial situation, the potential operators that can be applied to the problem and some goals state that needs to be met. After going through the available resource allocated for the task and the corresponding time constraint (if any, the planner produces the sequence of actions to achieve the goal.

Some planners work **offline**, which means they only generate plans and do not receive any feedback about the execution of the plan nor the state of the environment where the plan is executed. The generation of a plan is only based on the formal description of the environment. The state of the system at the time of executing the plan is assumed to be adequately modeled. Hence, a change in a state prior to plan execution is not taken into consideration. Thus, the domain model is usually robust enough to take care of the gaps or differences between the conceptual model and the real world. An example of offline planning is the generation of a regional advisory plan for traffic operators by a planner to solve an environmental problem using declarative knowledge of traffic strategies and policies within such environment. In this case, the plan generated has to be readable and interpretable by the user for decision making (by following the action sequence in the plan).

However, in a domain where the difference is more than what the planner can handle (in a non-deterministic environment) then, the planner will need to have constant feedback from the effect of its actions on the domain environment. Thus, **online** planning is usually employed in this situation. An example of online planning is an intelligent motion robot that uses AI planning for decision making. Such robot needs to continually sense, interprete and generate execution plans to change his state and the state of the world which it finds itself.

2.1.1 Planners Coverage

The coverage of a planner depends on the classes and complexities of problems that can be solve by the planner. Most planners have the ability to solve varieties of problems while some are tailored towards specific real world problems. A brief comparison of planners, based on their coverage area, will be explain in the next section with emphasis on domain *independent planners*. This is because most planners are domain independent and only few planners are tailored towards specific real world problems. Planner

coverage area can be categorised into three forms:

1. Domain Specific Planners
2. Domain configurable planners and
3. Domain Independent Planners

Domain Specific Planners These planners are tailored to work in a specific domains and might not work in other domains unless major modifications are made to the planning system. The planner in this categories uses specific information and knowledge from the domain to boost the planner performance and enhance the quality of plans generated at every instance of time. This is achieved by constraining the reasoning of the planner to search into a small part of search space. Thus, instead of searching through all forms of unnecessary states transitions to find a valid plan, the searching is restricted with the use of domain specific heuristics built into the planner code. These types of planners are very effective in creating plans for the target domain though, it might not be useful in any other planning domain; this might require building entirely another new planner. For instance, a planner designed specifically to solve problems in an urban traffic control domain cannot be use to solve problem in machine calibration domain. Example of planning in this category is HSTS [129].

Domain Independent Planners For several year, automated planning has been dominated with research on domain independent planning approach. This was facilitated by the creation of domain independent planners, show cased at every International Planning Competitions (IPC) [1]. These planners are meant to solve problems in varieties of domains. Example of planner in this category is LAMA planner [122].

Domain-Configurable Planners In domain reconfigurable planning, the planning engine is domain independent with some input of domain specific knowledge to guide

[1] http://ipc.icaps-conference.org

the search space. This makes it possible for the planner to work effectively in a new domain by providing additional information of the new domain to the planner functionality.

2.1.2 Planner Coverage Analysis

The above planner coverage could be compared based on three things:(a) Effort to configure the planner for a new domain.(b) Planner performance in a giving domain and (c) Coverage across many domains.

In terms of the effort needed to configure the planner for a new domain, domain specific planning requires the most effort to reconfigure for a new domain, because of the huge effort needed to build a new planner from scratch. Domain-independent planner on the order hand, do not need to be rebuilt provided the new domain satisfies all the restrictions of the planner. A domain-configurable planner requires encoding new domain descriptive information as part of the planning system, but does not need the creation of an entire new planner from scratch. Hence, domain configurable planners are somewhere in-between domain specific planners and domain-independent planners.

In practice domain independent planners have large coverage area compare to domain specific planner, this is because the later typically works with only one planning domain. On the other hand, domain specific planners have good and most effective performance when properly built. This is because, it is easy to encode domain specific problem directly in the planner.

Domain independent planner has the least level of performance, since it will not take advantage of any specific property of the domain.

As a result of the above analysis, a domain specific planner is more suitable to solve urban traffic domain problem and this thesis is a step towards realising that goal. As a result of the above analysis, a domain specific planner is more suitable to solve urban traffic domain problem and this thesis is a step towards realising that goal.

Domain Qualities	High	Average	Low
Planner Performance	Domain Specific	Domain Configurable	Domain Independent
Domain Coverage Area	Domain Configurable	Domain Independent	Domain Specific
Creation Effort	Domain Specific	Domain Configurable	Domain Independent

Figure 2.4: Table of Planner Coverage Analysis

2.2 Planner Algorithms

It is evidenced that even in very simple problems, the number of visible states in a planning problem could explode and be many orders of magnitude greater than the number of particles in the universe! Hence, it is impossible in any practical sense to list all of states in a giving planning problem explicitly. As a result of this, there is increase in the need for potent implicit representations that depict the necessary subset of a planning state in a more compact way that can be searched effectively. There arises several planning algorithms to solve planning problems. Some of the approach used for solving planning problems includes but not limited to:

- Plan-Space Planning
- Planning Graph
- State-Space Planning
- Problem Translation and
- Combination of Algorithms

Plan-Space Planning

This is a planning algorithm approach, whose nodes are set of partially-instantiated operators with some constraints. these constraints are increased until a plan is found. Plan-Space Planning searches through graph of partial plans such that, each node represents partially specified plans; each arc represents plan refinement operations, and the

solution is a set of partial-order plans. The idea behind this technique is to plan for a set of goals $g_1, ..., g_k$ by having a separate plan for each of the individual goal, while maintaining various bookkeeping information for detecting and resolving interactions among the plans for the realization of individual goals. It is a bit slower that some other approach, but new heuristics are now invented to improve it speed [36].

State-Space Planning

This is the algorithm approach that has been used in some planners that are successful IPC. Though, its usage in the past had been minimal, because of poor heuristic to guide the search. A breakthrough was made when it was realized that, heuristic values could be computed relatively quickly by extracting them from relaxed solutions [34; 87; 147]. In State-Space Planning, the search space is a subset of the state space such that, each node represents a state of the world; each arc represents a state transition, and the plan corresponds to the path through the search space. This is the basis of planning algorithms such as Fast Forward [74] planner and now used by most state of the art planners.

Planning Graph

Planning graph is another planning algorithm whose approach is useful in extracting heuristics for planners. A planning graph is of several levels such that for each n level, level n includes every action a such that at level $n-1$, the preconditions of a are satisfied and do not violate certain kinds of mutual exclusion constraints(mutex) of all the literals in that level. The literals at level n include the literals at level $n-1$, plus all the effects of all additional actions at level n. As a result, the planning graph represents a relaxed version of the planning problem in which several actions can appear at the same time even if they conflict with each other. In a planning graph, it is possible to eat your cake and still have it. This is because, 'eat your cake' and 'having your cake' could be included in the same graph expansion level.

The basic planning graph algorithm is as follows:

For $n = 1, 2...$ until a solution is found, create a planning graph of n levels. Do a backwards state-space search from the goal to try to find a solution plan, but restrict the search to include only the actions in the planning graph.

Considerable number of planning algorithms had been created out of planning graph [131], this is due to its ability to compute in a polynomial amount of time (that is, comparatively quicker). It seems to run faster than the plan-space planning algorithms. The efficiency of the backward search is noticeably improved, because of the backward search restriction to operate within the planning graph.

Problem Translation

This technique involves the conversion of planning problem into another kind of combinatorial problem, example is a satisfiability of integer programming, which has an already existing efficient problem solvers. The translated problem is then solved with this solvers and the solution found from this solvers (integer programming or satisfiability) is then translated into a plan. One of the major drawback of this approach is the ability to cope with large domains with enormously large systems of constraints. For instance, a function with many argument k would generate a collection of atoms of size exponential to k [126]. However, this technique had led to several satisfiability planners that could solve complex problems [44; 80; 81; 143].

Planning in Non-deterministic and Probabilistic Domains

Researchers have been able to extend planning techniques to work in nondeterministic and probabilistic planning domains. Non deterministic planning domain [2] are problems whose state of the world are not completely known to the planner. This types of problem shares some similarities with probabilistic planning domain, except that, the outcomes of a nondeterministic domain might have no probabilities attach to them. In,

a probabilistic domain, there are multiple possible and valid outcomes in an environment that is partially observable or non deterministic, the probability of each possible outcome has to be taken into consideration [86; 138].In this case, the most favourable outcome has to be selected and considered in every step of the planning process.

Combination of Algorithms

Planning algorithms could be derived from a combination of existing approach. Some planners use one or more combination of the above algorithms. For example, a state space planner might use satisfiability approach to optimise numeric fluent and a plan space planner might employ a Markov decision processes to choose favourable outcome in a probabilistic domain. A planner(portfolio) might also combine several independent planners with different algorithms in a single planner for the purpose of solving varieties of problems [137].

2.3 Planning Modes and Capability

Planning has evolved over the years from the basic simple logical planning(Classical planning) to the complex stochastic and continuous planning. This means that the ability of planners to reason with increasingly complex real world problems is a field of knowledge that is still explored by most researchers in the field of automated planning. This sections explains the various types and modes of planning platforms with their associated approach towards problem solving.

2.3.1 Classical Planning

This planning approach [68] involves abstraction of environmental information and problems into a declarative logical representation model. The abstracted model is meant to represent the environment without compromising any necessary detail for the environment to function effectively. Classical planning has some restrictive assumptions

which represents the beliefs of the state of the world. Some of the assumption that could be made for a given system are as follows:

1. Assuming that the system has a finite set of states
2. It is fully observable. This means that, one has complete knowledge about the state of the system.
3. It is deterministic, which means that every state and event or action is deterministic. Whenever an action is applied to a state or event, it makes a deterministic system transit to another deterministic state or event.
4. It is a closed system. This means there are no exogenous events.
5. The goal state is explicitly specified as a goal state or a set of goal states that must be attained, irrespective of any constraints on state trajectories, state to be avoided and utility functions.
6. Solution plans to a planning problem are often linearly ordered sequence of actions.

Classical planning is the simplest form of AI planning in which the set of events E is empty and the transition function γ is deterministic (i.e. $\gamma : S \times A \rightarrow S$). The least of which is the STRIPS approach to planning [53]. *STRIPS* planning involves finding a sequence of operators in a space of world models to transform a given initial state to a goal state where the goal holds. The planner increment the plan with additional actions trying to create an effective transformation from the initial state to the goal state. The operation of the STRIP planning is in the form of $OP = (pre, del, add)$, where 'pre' means the precondition that needs to be valid immediately before the operator is applied, 'add' and 'del' are the set of literals that are added or deleted from the present state after the operator is executed. The instantiated operator which is being added to

a plan is called action. However, the expressivity of the STRIPS actions model is very limited, since it does not allow to abstract knowledge in different levels.

Some desired application for planning and scheduling typically have the following characteristics: a continuous changing world; multiple agents (both cooperative and adversarial); uncertainty of the state of the world; communication with external sources (sensors, databases, human users) to get information about the world; overlapping actions; time constraints and numeric computations involving resources, probabilities, spatial relationships and geometric. Unfortunately, classical planning does not possesses any of these characteristics.

2.3.2 Metric Planning

Real world situations often involves quantities and numeric values, whose representations and manipulations are not depicted in classical planning. The need to express numeric quantities of entities within a domain representation led to the development of metric planning. Numeric planning introducing the use of numeric optimisation function which is stated within the problem descriptions file. For instance, an objective to minimise travel times can be used in domains where the traveling value is a numeric variable. An optimisation function could also be in problem file which includes the special variable "total-time", representing the total duration of execution (makespan) for the plan.

One line of approach to metric planning is to represent numeric values in planning domains as discreet numeric variables. Action can have conditions expressed in terms of this numeric values and have effect that act upon them. The heuristic guidance for such scenario can be provided by [74] by introducing an extension to the relax planning graph(RPG) heuristic [74], the Metric RPG heuristics generates a relaxed plan that support the computation of problems involving discrete numeric change. This is similar to RPG by performing- forward searching while ignoring the delete effect of actions.

However, in this case ignoring decrease effect does not always relax the problem, because a variable could hold a value more than the given constant in the condition. Thus, the upper and lower bound on the values of numeric variables are computed and used during the extension of reachability analysis. A later alternative to the Metric RPG was proposed by [31] in LPRPG using a linear programming to incrementally capture the interaction between actions. This approach provides a more accurate heuristic guidance for metric domains especially in problems where metric resources must be exchanged for one another in order to complete a solution. The setback with this approach is that, it is only restricted to action with linear effects compared with Metric-FF [73] which is a general approach. Though, the optimisation of metric function is a bottle neck for planners, but some planner still attempt to optimise these functions such as LPG planner [64].

2.3.3 Temporal Planning

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 has had several contributions prior to the introduction of PDDL2.1. The introduction of chronicles [91] by IXTeT pioneered the use of many temporal representation in planners including single temporal network and linear constraints. It consists of temporal assertions and constraints over a set of state variables, and timelines which defines chronicles for single state variables. Timeline representation also used by some planners that follow a different path of advancement than that led by the PDDL family [9; 25; 129].

One line of approach to temporal planning is the use of Temporal Graph Plan(TGP)by [130]. TGP allows constant durations to be attached to actions which require preconditions to be true for the entire duration of the action. The effect of the actions instantaneously becomes true at the end of the action execution. The semantics is that, the values of variables within actions should be observed during the interval of execution of the action and plans should be conformant with all the changes in values of these variables during the action interval. However in TGP, actions are treated as undefined and inaccessible during the implementation interval. This setback in TGP was solved with the temporally extended version of Graphplan planning graph to cope with temporal constraints. A constraint propagation approach was developed to handle the temporal heuristic effect of this type of planning problem by [70; 141]. Most temporal planners such as MIPS-XXL [47] still makes use of this restricted TGP semantics through the exploitation of simplification of PDDL 2.1 encoding known as *action compression*. Though applying this compression can leads to incompleteness which could result in the failure of a planner to solve certain problems [30]. Some planners are able to reason with the PDDL2.1 'start' and 'end' semantics, as opposed to relying on a compression approach such as VHPOP [149] - a partial-order planner.

SAPA planner [43] is one of the foremost forward-search planners to solve temporal PDDL2.1 problems. It works by building a priority queue of events. This allows SAPA to reason with concurrency and to solve certain problems which required concurrency in execution. This is achieved by queuing the end point of a durative action to a time in future at which it needs to be executed. Thus, the planner has a choice to start any new action, but also a special wait action which computes and advances time to the next entry in the queue, and the resultant action is executed. It also uses multiple integer programming to optimise plan metrics and sequence. It uses the FF heuristics and helpful actions to guide search. A drawback with SAPA is the fact that its search space does not include all necessary overlapping of actions to achieve a complete search. This is as a result of the waiting time approach for the priority queue, because it is

hard to determine explicit waiting time if the next interesting time point depends on the interaction of actions that have not yet even been selected.

The Crikey family on the other hand [30; 32], employed a different approach to temporal planning compared with SAPA. It uses the relaxed graph plan [74] with an extension of some guidance from the temporal structure of the plan, to guide a forward search. The planners use a Simple Temporal Network(STN) to model and solve the temporal constraints between the action end points as they are accumulated during successive action choices.

The Temporal Fast Downward(TFD) planner [48; 72] uses an approach that is a slight enhancement of the compressed action model used by TGP, allowing some required concurrency to be managed. Both TFD and SAPA face the same problem in situations in which an action must be started sometime after the last happening, but before the next queued event: neither planner includes this choice in its search space.

There are other temporal planners that use SAT solvers, such as [76], that was developed using 'planning-as-SATisfiability' paradigm. This uses a combination of Graphplan to SATisfiability approach 'Graphplan-to-SAT' encoding. This approach is especially suitable for problems where an appropriate time increment can be identified. Most of the temporal planners mention so far is restricted to the computation and management of discrete changes. Continuous duration changes cannot be handled by these planners.

2.3.4 Continuous Planning

Continuous planning extends temporal planning by considering constantly changing state variables with respect to time. Continuous planning in domains which are represented with rich notations has long been a great challenge for AI [19]. For instance, changes occurring due to fuel consumption, continuous movement, or environmental conditions may not be adequately modelled through instantaneous or even durative actions; rather these require modelling as continuously changing processes. The combination of time dependent problems and numeric optimisation problem create a more challenging and hard task of time-dependent metric fluents.

One of the earlier work that involves planning with continuous processes includes the Zeno system [113]. In this case, processes are described using differential equations rather than as continuous update effects, so that simultaneous equations must be consistent with one another rather than accumulating additive effects. McDermotts OPTOP system [99] is another early planner to handle continuous processes. A forward search planner that avoids grounding the representation using a unique approach to generate heuristics(relaxed plan estimates of the number of actions required to achieve the Goal) from a given state.

TM-LPSAT [145] was built upon the earlier LPSAT [145]. It was the first planner to implement the *PDDL+* Semantics [58] and an addition capability of handling variable durative actions. This includes durative actions with continuous effects and duration-dependent end-effects. It uses PDDL+ semantics to compile a collection of SAT formulas from a horizon bounded continuous planning problem, together with an associated set of linear metric constraints over numeric variables. The compiled formulation is passed to a SAT-based arithmetic constraint solver, LPSAT [4]. The SAT-solver parses triggered constraints to the LP-solver, if there is no solution the horizon is increased and the process repeats, otherwise the solution is decoded into a plan. The novelty of TM-LPSAT lies in the compilation of the PDDL+ semantics and decoding of the SAT

solver into a plan, since both solvers are well-established systems.

The theory behind TM-LPSAT is very promising with the capability of solving large class of problem with varied continuous dynamics. However, empirical data suggests that the planner requires large CPU time to generate plan and unable to solve problems requiring plans of more than a few steps.

Kongming [94; 95]is another domain dependent continuous planner that solves a class of control planning problems with continuous dynamics. The language used is a version of PDDL2.1 extended to enable dynamics to be encoded. It is based on the building of fact and action layers of flow tubes, using the iterative plan graph structure of Graphplan algorithm [17]. As the graph is expands, every action produces a flow tube which contains the valid trajectories as they develop over a period of time. Reachable states at any time can be computed using the state equations of the system starting from a feasible region, and applying actions whose preconditions intersect with the feasible region. Kongming translates a planning problem into Mixed Logical-Quadratic Program (MLQP) using the plan-graph encoding with the continuous dynamics of the system. The planners metric objectives function can be defined in terms of quadratic function of state variable. Time is discretised to support state update within the plan - successive layers of the graph are separated by a constant and uniform time increment.

UPMurphi is another planner [116] that uses PDDL2.1 to reasons with continuous processes on a batch chemical process [115]. It alternatively refines a discretisation of continuous changes until the solution to the discretised problem validates against the original problem specification. UPMurphi starts by discretising the continuous representation of the problem. Specific values within feasible ranges are taken as actions. Given rise to several version of each actions. The current discretisation is then used to explicitly construct and explore the state space. Plans are constructed in the form of planning-as-model-checking paradigm [28] with no heuristic to guide the search. When a plan is found, it is validated against the original continuous model, using the plan validator [56]. If it fails to find a plan at one discretisation, it alterate again at a

finer grained discretisation. Successive refinements lead to ever denser feasible regions, which might be increasingly complex to construct.

COLIN [33] is a forward-chaining heuristic search planner that uses the temporal semantics of PDDL2.1 and also capable of reasoning with continuous linear numeric change. It combines forward chaining search of FF to prune state, with the use of a Linear Program (LP) to check the consistency of the interacting temporal and numeric constraints at each state. The Temporal Relaxed Planning Graph heuristic of CRIKEY3 [32] is also extended to support reasoning with continuous change. A mix integer programming is used for post processes to optimise the timestamps of the actions in the plan.

There are other approaches to continuous reasoning which might not be listed in this thesis. An example of such approach is to model continuous resource consumption and production in terms of uncertainty about the amount consumed or produced [102]. This lead to the construction of a Markov Decision Process (MDP) consisting of hybrid states.

Automated Planing in continuous process, allows us to model actions, process and event [58] as part of domain operators. One line of representation of continuous domain is to use *PDDL+* [58], an extension of PDDL2.1 to represent exogenous events and processes as well as durative actions. It is assumed that an action can causes a discrete state change which might trigger a continuous process. This process continues over time until an event is initiated leading into a new state. While in that new state another action might be taken which could trigger a new process or a new action.

The following additions extends the class of temporal-metric problems to temporal-metric-continuous problems:

- a set of linear continuous numeric effects is added to the component of each action a. This corresponds to the PDDL2.1 effect $(increase(v)\ (*\ \#\ t\ k))$ which denotes an increases in v at the rate of k per unit of time.

- $eff_{\vdash}n$ and $eff_{\dashv}n$ may additionally include durative parameter $?duration$ for both *start* or *end* effects of actions. This denotes the duration of the action, and could be written as:

$$\langle v, op, w \cdot v + k \cdot (?duration) + c \rangle, \text{ such that } op \in \{+=, =, -=\}, c\ , k \in \kappa$$

  where w is a vector of constants and $?duration$ may have a constrain value that lie within a range of values, e.g. $(?duration \geqslant v1) \bullet (?duration \leqslant v2)$, for some numeric variables *v1* and *v2*.

The relationship between time and numbers is more complex in a continuous planing problem than in temporal-metric problems. The extension to the action component allows the value of a variable v to depend on the period time elapsed from the beginning of the continuous effect acting upon it. The second extension the durative effect implies that, $?duration$ do not always need to be fixed, since the the value of variables can now depend on the duration assigned to the action.

2.4 Planning Representation and Language Semantics

There exist planning representation and ontologies for representing problems and domain information in a way that is comprehensible to a planner. An instance of this is the Stanford Research Institute Problem Solver (STRIPS) approach to planning [53]. *STRIPS* represents a model of the world as an arbitrary collection of first-order predicate calculus formulas. For example 'girl(rofia)' means *Rofia is a girl*.

Different representations has evolve over time, depending on the nature of the implementation and the type of planner that can interpret the language semantics. For instance, Object Centered Language(OCL_h)is developed for the acquisition of hierarchical domains [97][98] . It is based on the idea of engineering a planning domain so that the universe of potential states of objects can be defined first , before operator definition. It is object centred because it deals with object and it relationship with dynamic or static states. Another example is PPDDL [148] (an extension of PDDL) or RDDL [125] for modelling conformant planning problems.

However, planing Domain Description Language (PDDL) [100] is currently the dominant language within the planning community for describing planning domains and problems. This is due to the need to have a uniform language within the planning community for the creation of benchmarks(domains) in the International Planning Competition (IPC)[1]. PDDL is the closest language to the representation used in this thesis, thus the next section gives an overview of PDDL semantics.

Planning Domain Definition Language (PDDL)

PDDL can be understood as a language, with a lisp like syntax, for describing planning domain and modelling abstractions of real world problems. It was developed from STRIPS in 1998 by Drew McDermott with the aim of being a specification of planning models and later improved upon and show cased at every International Planning Competitions (IPC). It is used due to the need for representation and exchange of domain model within the planning community [100].

It is also important to know that PDDL has different family of languages such as PPDDL, RDDL as mention in the previous section. Figure 2.5 shows an example of a painting domain expressed in PDDL. This domain required two robots to paint five different rooms at different time duration and having some constraint with the paint pots. They need to get and move all the require painting material with them before painting

[1]http://ipc.icaps-conference.org

```
(:durative-action wet_brush
   :parameters
        (  ?ppot - paintpot ?rob - robot
             ?bru - paintbrush ?bruc - paint
             ?potc - paint ?r - room )
   :duration
        (= ?duration 4)
   :condition
        (and
        (over all(in ?rob ?r))
        (over all(rob_hold ?rob ?bru))
        (over all(at ?ppot ?r))
        (over all(colour ?ppot ?potc))
        (at start(colour ?bru ?bruc))
        )
   :effect
        (and
        (at start(not(colour ?bru ?bruc)))
        (at end(colour ?bru ?potc))
        )
 )
```

Figure 2.5: A excerpt of a PDDL painting domain model showing the action to *wet_brush* with paint

and there is limitation to the number of rooms the paintbrush can paint. The generated plans for the execution is shown in Figure 2.6. .

Levels of PDDL

An advancement of the STRIPS representation to deal with numeric fluents and temporal actions can also be represented in PDDL2.1 [57]. PDDL2.1 has different levels with level 1.0 restricted to propositional models. Level 2.0 introduces numeric fluents, to create reasoning about numeric values in domains. Though, the use of numeric fluents at level 2.0 is restricted to basic numeric operations, but it was able to solve some basic numeric problems compared to level 1.0.

```
 Time: (ACTION) [action Duration; action Cost]
 0.0000: (PICK JOHN BRUSH2 ROOM2)[D:2.0000; C:0.1000]
 0.0000: (MOVE ALAN ROOM1 ROOM2) [D:3.0000; C:0.1000]
 2.0000: (DIP_BRUSH_IN PAINTBUCKET2 JOHN BRUSH2 WHITE BLUE ROOM2)[D:4.0000; C:0.1000]
 3.0000: (PICK ALAN PAINTBUCKET1 ROOM2) [D:2.0000; C:0.1000]
 5.0000: (MOVE ALAN ROOM2 ROOM3) [D:3.0000; C:0.1000]
 6.0000: (PAINTING JOHN ROOM2 WHITE BLUE BRUSH2 PAINTBUCKET2)D:40.0000; C:0.1000]
 8.0000: (DROP ALAN PAINTBUCKET1 ROOM3) [D:2.0000; C:0.1000]
 10.0000: (MOVE ALAN ROOM3 ROOM2) [D:3.0000; C:0.1000]
 13.0000: (PICK ALAN BRUSH1 ROOM2) [D:2.0000; C:0.1000]
 15.0000: (DIP_BRUSH_IN PAINTBUCKET2 ALAN BRUSH1 WHITE BLUE ROOM2)[D:4.0000; C:0.1000]
 19.0000: (MOVE ALAN ROOM2 ROOM3) [D:3.0000; C:0.1000]
 22.0000: (PAINTING ALAN ROOM3 WHITE BLUE BRUSH1 PAINTBUCKET1)[D:30.0000; C:0.1000]
 46.0000: (DIP_BRUSH_IN PAINTBUCKET1 JOHN BRUSH2 BLUE RED ROOM2)[D:4.0000; C:0.1000]
 50.0000: (PAINTING JOHN ROOM2 BLUE RED BRUSH2 PAINTBUCKET2)[D:40.0000; C:0.1000]
 50.0000: (DROP ALAN BRUSH1 ROOM3) [D:2.0000; C:0.1000]
 90.0000: (MOVE JOHN ROOM2 ROOM3) [D:3.0000; C:0.1000]
 93.0000: (DROP JOHN BRUSH2 ROOM3) [D:2.0000; C:0.1000]
 95.0000: (MOVE JOHN ROOM3 ROOM4) [D:3.0000; C:0.1000]
 98.0000: (MOVE JOHN ROOM4 ROOM5) [D:3.0000; C:0.1000]

Solution number: 1
Total time:      0.02
Search time:     0.00
Actions:         19
Execution cost:  1.90
Duration:        101.000
Plan quality:    101.000
```

Figure 2.6: Painting Plan generated by lpg-td-1.0 planner for two robots painting several rooms with time and resource constraint

Level 3.0 captures reasoning with time by introducing durative actions.The problem is still a tuple $\langle A, I, G \rangle$ with a modification to the action declaration. The durative action defines a model of time to reason with the start and end of actions. The conditions in a durative actions can be defined to either hold at the start, at the end or for the entire action execution(invariants). Fluent are assigned instantaneous values at the initial states or by an action. This could be definite value, or an increase or decrease by a value; or the use of basic arithmetic operators $(+, -, *, \div)$ to modify fluent base on arithmetic expressions. The durative action imposes a upper and lower bound constraint on the action operator. The declarative language for the constraint bound is dictated by a numeric expression with a language declaration similar to that of numeric preconditions; this could be equal in the case of fix durations. This level encompasses the numeric characteristics of level 2.0 given rise to the challenges of manipulation time and numerics within planning problems. Numeric conditions used in *dur*, $pre \vdash$, $pre \longleftrightarrow$, $pre \dashv$ and G can be expressed in the form:

$$\langle f(v), op, c \rangle, \text{ such that } op \in \{>, \geqslant, =, <, \leqslant\}, c \in \kappa$$

where v is a vector of metric fluents in the planning problem, f(v) is a function applied to the vector of numeric fluents and c is an arbitrary constant.

Also, numeric effects used in $eff \vdash$ and $eff \dashv$ can be expressed as:

$\langle v, op, f(v) \rangle$, where $op \in$ numeric operators such as $\{* =, + =, \div =, =, - =\}$

Each operator comprises of $\langle dur, pre \vdash, eff \vdash, pre \longleftrightarrow, pre \dashv, eff \dashv \rangle$, where:

- $pre \vdash$ is the start condition of action a at the state in which a starts and $pre \dashv$ is the end condition of a at the state in which a ends. This two condition must hold for a in both situation.

- $eff\vdash$ is the start effects of a at the state in which a starts and $eff\dashv$ is the end effect of a at the state in which a ends. Both $eff\vdash a$ and $eff\dashv a$ updates the world state according to these effects. Thus, a given collection of effects $eff_z, z \in \{\vdash, \dashv\}$, consists of:
  - eff_z-, propositions to be deleted from the world state
  - eff_z+, propositions to be added to the world state
  - eff_zn, effects acting upon numeric fluents
- $pre \longleftrightarrow$ is the invariant conditions of a which must hold at every point in the open interval between the start and end of a.
- *dur* refer to the special parameter $?duration$, which denotes the duration of a. It is the the duration constraints of a, compute on the basis of the world state in which a is started, and restraining the length of time that can pass between the start and end of a.

Level 4.0 PDDL 2.1 expresses linear continuous change of numeric fluents. This represents a continuous change in numeric value, according to some specified numeric functions, over the entire period of execution of an action. A differential equation of the form $\frac{dp}{dt} = f(P) + c$, specifies the effects on a variable p where P is the vector of state numeric variable, $f(P)$ is a mathematical function over these numeric variables, and c is a constant.

Level 5.0, also referred to has *PDDL+* proposed a realtime temporal model that includes actions, exogeneous events, and autonomous processes. This is an extension of level 4.0 with an introduction of process and event operator. This type of problem has been dealt with by some planners [116; 145], yet this nature of problems is still a challenge in the planning community especially in complex continuous domains with multi-modal resources. The input level for the planner implementation in this thesis is at the level of *PDDL+* with few modifications that will be discussed in Chapter 6.

2.5 Automated Planning Applications

Automated planning application cut across several disciplines and different applications ranging from business and commerce to entertainment entrainment industry. This section summaries some of the exiting implementation of Automated planning towards solving real world problems.

2.5.1 Software Development

In software development, AI planning techniques can now be used to support automatic generation of evolution paths [8], when developing a theoretical framework to help software architects make better decisions when planning software evolution. Goal-driven automated composition of software components, an important problem with stream processing systems is now managed with the use of Hierarchical Task Network (HTN) planning for the composition of stream processing applications [133].

2.5.2 Space System

The need to maximise resources; optimise constraint and cost of operation have led NASA and others to utilise AI planning in other to move towards total on-board autonomy [26][22]. Also AI planning was is used to create an automated system that

support the management of the (Technology Demonstration Payloads)TDP operations of a spacecraft [121] by coordinating all the different, possibly conflicting, payload operations of the spacecraft. Another recent development in the field of space exploration systems is the use of AI planning approach to manage the solar arrays that powers the International Space Station (ISS) [61]. It uses constraint management and automated planning capabilities to manage the solar arrays that powers the International Space Station (ISS). This approach automatically generates solar array operations plans by reasoning about constraints, finding optimal array configurations subject to these constraints and solution preferences.

Health Sector

AI planning enables the automatic generation of patient-tailored treatment plans. This approach uses multiple Computer Interpretable Guidelines (CIG) languages [14; 39] as a basis for expressing computational knowledge in clinical decision support for the treatment of patients with specific conditions[51]. Another solution in the health sector is the use of AI planning to create a planning-based socially intelligent agent that interact and helps children with Autism Spectrum Disorder to acquire and develop some social communication skills [15].

2.5.3 Business Management and Commerce

The creation of domain independent planners and execution environment that is suitable for binding business applications [134] increases the curiosity of researchers to investigate the use of AI planning to enhance business management. AI planning approach was later used to generate a working plans in organisation processes by translating the Business Process Modelling Notation (BPMN) into a model that can automatically execute, monitor and optimise planning process and resources, base on the desired plans [52]. In addition, action base planning language [3] is used to specify a business process from the banking domain that is representative of an important class of business processes of practical relevance, to automatically solve reasoning tasks that arise in specific business context.

AI planning and scheduling approach is use to help business planners and controllers, in one of the largest limousine cab company in the world, to handle and manage fleet while maintaining high quality service. This project helps to maintain effective and efficient delivery of quality service to customers despite continuous expansion of resources and constraints[27] within he business. Furthermore, automated planning is use to optimise control of integrated devices and appliances with the objective of saving energy while the comfort and productivity of the occupants in an office environment is preserved [62]. This approach uses simple sensors from the office environment; process context information from ontology-based occupant activity recognition and an automated planing system to control appliances. This strategy has a recognising accuracy of about 80 percent when tested in an actual living lab. A similar approach used AI planning to reduce the energy consumption of network systems (servers, network equipment, air-conditioners, etc.) within an office space [5].

Web services on, the other hand, have attracted significant interest as a low-cost and flexible technology alternative for delivery of on-demand business processes, enabling intra-organizational systems amalgamation, and in the long term, development of dynamic inter-organizational process networks that is now transforming electronic commerce. PORSCE II [71] is an example of an integrated system that uses automated planning to performs automatic Semantic Web service composition. This is achieved by dynamically translating the Web service composition problem into a solver-ready planning domain and problem, the desired composite service solution is obtained from a domain independent planning system and is translates back to Web service terms. Another example is the area of tourism is the use of automated planning to customised tourism preferences in web services application [23][109] and the use of both planning and constraint programming approaches to compute optimal solutions in problems involving the designing of personalised museum visits [16].

2.5.4 Education Sector

In order to make the learning process more efficient, attractive and accessible to students, customise learning design for student was implemented using AI planning [24]. This helps to create an automated planning and monitoring system that is able to build up a learning path for a specific student which is adapted to its user profile. Timeline-based planning is also use in dynamic training environment for crisis managers [37]. This helps a trainer to create and deliver engaging and personalized training lessons for decision making skills within a crisis management domain.

Also, an hybrid planning formalism is now used to create a domain independence assistance which combines a number of planning and interaction components to realize advanced user assistance [13]. This is aimed at assisting users to achieve certain complex task, such as assisting a user in the assembly of this home theater.

2.5.5 Game

The vitual world of Games have attracted significant interest in the entertainment industry. An offline planning approach is used to optimise the global plotlines structure in a role-playing game based on knowledge of players preference [93]. This approach uses a partial order planning and a hierarchical decomposition of the main plot line into several quests comprising of a primitive task and associated rewards to achieve the semantic meaning of the original plot. An adaptive algorithm was created that can alternatively search using partial order planning to optimise the selection of the preferred story line by its ability to create, repair and eliminate causal links between plans (additive and subtractive improvement mechanisms).

2.5.6 Industrial Control

Jana and Daniel where able to use information gotten from a learned lift traffic pattern in a controlled algorithm to prioritise and control the way different floor are served during lift operation. Thus, destination control NP-Hard online problem [83] was modelled in terms of a planning and scheduling problem as well as a constraint satisfaction problem [107]. The allocation algorithm was address as combination of static, offline optimisation problem and a dynamic, online allocation problem. The optimisation metric was based on the combination of waiting and journey times. It was able to augment a fast forward technique with a depth-first branch and bound search algorithm for the offline process.

Planning is also used to generate detailed procedures in industrial manufacturing processes [124] by which work fragment pieces are converted from the raw material form to the desired form. The use of AI planning for machine tool calibration was another burgeoning project that aim at reducing calibration error while reducing the total time taken for the entire tool calibration process [111]. The performance evaluation shows that automatically generated plans are faster and more error-free than human generated plans. This research was later improved to reduce uncertainty of measurement in machine tool calibrating with the use of state-of-the-art artificial intelligence planning tools to generate optimal calibration plans that reduces uncertainty of measurement in machine tool calibration [112].

Similarly, a substantial reduction in the cycle time of remote laser welding for automotive industrial operation has recently been proposed by introducing a new model and algorithm for task sequencing [85]. This is achieved by modelling the problem as a travel sales man and path planning problem; the problem is them solved with a tabu search algorithm

The use of robotics in industrial control is also supported by AI planning technology. An example of robotic in industrial application is the coordination of fleets of autonomous, non-holonomic vehicles. This multi-robot approach [29], can jointly plan for multiple vehicles to generates kinematically feasible and deadlock-free motions. Another contribution is the application of planning to enable task-based social interaction between robot and multiple human agent in a bartending domain [118]; the the aim of serving different customers irrespective of there preferences.

Another development in the field of power industry is the use of AI planning to optimise power flow in distribution in power grids. AI planning solves the electricity network balancing problem which is aim to ensure that the electricity demands of the consumers are met by the committed supply [119]. This is achieved by using action planning to find feasible state trajectories in a discrete state space from an initial state to a state that satisfies the goal condition while allowing the complex numeric changing

parameters to be handle by an external module(connected to the planning engine).

In the transport sector, AI planning is used for intelligent transportation network. For instance,multi-agent planning approach is used to implements a decentralized, schedule-driven traffic signal control [18]. Scalable Urban Traffic Control(SURTRAC) computes a schedule that optimizes the flow of currently approaching traffic at each intersection independently (and asynchronously). The resulting schedule is used to decide when to switch green phases and to indirectly influence the schedule of neighbouring intersection. This helps to provide real-time responsiveness to changing traffic conditions and coordinated signal network behaviour. This thesis is also an effort towards applying AI planning to transport which will be discussed subsequent chapters.

2.6 Summary

This chapter gives an overview of various terminologies in Automated Planning used by different chapters in this thesis. It explains the concept of Automated Planning as a branch of artificial intelligence and how it applies to real world problems. This chapter ends with an overview of some applications of automated planning in different field of knowledge. The next chapter introduces the basic concept of Urban Traffic Control.

Chapter 3

Introduction to Urban Traffic Control

The evolution of Urban Traffic Control (UTC) throughout the past century has been a continued effort to maintain and stabilise the increase in vehicle demand without violating the ever more complex policy objectives of transport systems. UTC has evolved since the creation of mobile entities for commuting people and goods with the use of traffic officers for signalisation, monitoring and control. The first generation of automated traditional controller uses fixed timing signals to control traffic at junctions. The second generation of traffic control used historic data to pre-configure traffic timing and control. The third generations use real traffic data for re-adjusting and choosing traffic signals with the use of detectors for the collections of real traffic data. The fourth generations were able to compute and manipulate data for forecasting traffic conditions for the creations of traffic control programs and strategies for optimal control of traffic situation. These trends continue, until the introduction of intelligent traffic control, which propose the use of artificial intelligent techniques combined with all other existing traffic approaches for real-time detection; traffic forecasting; traffic signalization incident detection based on the principle of integrated embedded and control system engineering [69]. The field of intelligent transport systems (ITS) is now honed in the twenty-first century with advances in efficiency, quality and manufacturability of sensing and communication systems for traffic control. Efficient UTC system has helped

to reduce congestion, increased economic efficiency and improved road safety and air quality [69]. Thus, urban Traffic management involves planning, monitoring and control of traffic flows with the aim of maximizing the effectiveness of the use of traffic infrastructure; ensuring reliable and safe operation of transport system while addressing environmental goals and policy objectives [55; 144].

3.1 Basic Component of Urban Traffic Control System

There are basic components that are fundamental to the implementation of urban traffic control system. These includes Road Traffic Network and Disturbances; Traffic Control Strategies and Rules; Traffic Operator(human and non-human) and Traffic Devices(controllers and sensors). This section gives a brief overview of these components.

3.1.1 Road Traffic Network and Disturbances

A real traffic network contains information about the state of a road traffic environment at any specific point in time. Changes to the state of a traffic network are under the influence of various disturbances. Disturbances are factors that affect the normal functioning of a road network. Disturbances could be planned or unplanned. An example of planned disturbance is a set of road works. This disrupts the normal flow of traffic for the 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. Unplanned disturbances, on the other hand, are factors that are not envisaged before there interruption of the traffic network. These factors include but is not limited to road incident and unexpected change in traffic demand and severe weather conditions.

Figure 3.1: Basic Component of Urban Traffic Control System

3.1.2 Traffic Control Strategies and Rules

Traffic Control Strategies and rules are extracted from the Traffic Management Policy and Guidance which is one of a series of policy documents that outline how a County Council manages, maintains and is developing transport infrastructure. Traffic Control Strategies and rules varies from county to county and changes over a period of time. A change in this rules could alter the metric and goals of any traffic control system. Traffic Management measures can include on-street parking controls, speed limits, HGV restrictions, direction signing, traffic calming, movement restrictions and pedestrian crossing facilities.

3.1.3 Traffic Operator

Controlling road traffic involves directing vehicular and pedestrian traffic within a network of connected roads; around a construction zone; accident or other road disruption. Thus, the major role of traffic operators is to manage the road network so as to ensure,

as far as possible, the safe and convenient movement of traffic. They also ensure the safety of emergency response teams, construction workers and the general public. This could be done manually or through the use of sensing and controlling devices within the network of roads. The traffic operator is empowered to influence the control of the entire system whenever there is genuine reason to alter the goal or metric of the system as a result of policy change. Human-machine interface is the communication interface between the traffic environment and a human operator. Traffic system allows the flexibility of being altered by a human operator. Human rules and desires are subject to changes, thus, the operator is able to influence the operating policy of the system by changing the goals and metric of the system when the need for changes arises, this might require re-installation of part or entire traffic component due to such changes.

3.1.4 Traffic Devices

Surveillance and sensing devises are embedded into some 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 a system or a traffic officer. For example in an urban area, existing traffic detection technologies includes but not limited to: Video Camera, Inductive Loops, Microwave and Infrared Sensor technology, Magnetic technology, Acoustic technology, Radio Frequency technology, Radar technology and Global Positioning System (GPS).

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 ranging 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. The control devices change the state of the road network to desires state and example is the traffic signal heads.

3.2 Urban Traffic Control Application

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 [69]. This section discusses few of the UTC control that are related to our research interest. This involves the use of different techniques or combination of two of more techniques involving model predictive control (MPC) to solve traffic problems.

Model Predictive Control is used to control junction-based subsystems within a traffic network [142]. Such that each junction has a different MPC controller that take its input from both control effect and control input of adjacent subsystems junctions. This approach makes it possible to handle larger network because of its distributed nature, thus making it suitable for complex urban networks.

A similar approach uses a rolling horizon for optimisation of traffic-responsive optimal signal split taking uncertainty into account [139]. In this case, a traffic control problem is formulated in a centralized rolling-horizon fashion such that the traffic prediction is influenced by unknown, but bounded demand and queue uncertainty. The objective function of a constrained minimax optimization formulation is minimised to obtain the green time combination of the traffic control.

Another application is the uses a generalise Model Predictive Control(MPC) framework to minimize network congestion when applied to both centralized traffic signal and route guidance systems [89]. The result shows that using a central control scheme may reduce the congestion inside a network while still achieving better throughput compared to that of other conventional control schemes.

Likewise, similar application also uses dynamic traffic assignment on urban traffic control problem to show the applicability of traffic models in cases that are an anomaly to the standard test network of similar research [103].

Model Predictive Control (MPC) framework can help to improve the network efficiency in urban traffic network by formulating signaling split control as an MPC prob-

lem [146]. This control strategy proof more efficient in relieving traffic congestion of a whole traffic network when compared to the conventional fixed time control.

Model predictive approach is also used to create a hierarchical control structure that combines the optimal coordinated model-based ramp metering control strategy AMOC (Advanced Motorway Optimal Control), with a local feedback ramp metering strategy ALINEA and a variation of it [84]. The coordinated ramp metering problem is considered as a finite horizon constrained non-linear discrete-time optimal control problem with a local feedback regulator based on downstream occupancy or density and flow measurements. The simulated result shows good promising scalability when tested on an Amsterdam ring road.

Traffic policy in urban areas can also be controlled with the use of MPC [128]. This involves the design of a trimodal model of urban traffic consisting of private vehicle, buses and Bus Rapid Transit (BRT). The approach is based on traffic-responsive Urban Control (TUC) strategy to present a predictive control model that minimizes a cost function to control urban traffic.

A customised linear prediction algorithm could be used to estimate future traffic intensities at different intersections within an urban traffic network [106]. A central controller could reduce the congestion level of traffic at such intersection by re-routing the vehicles and adaptively changing the signaling cycles based on the prediction. This approach is uses the Vehicular Ad-hoc Network (VANET) architecture with a novel communication scheme to propose a predictive road traffic management system (PRTMS). Performance improvement was recorded with this approach in terms of total journey time and waiting time of the vehicles.

Certain traffic control applications are agent-base. Agent-base technology involves the use of UTC programmes that can act and take decisions on behalf of human traffic officers. For example, intelligent agents exhibiting some aspect of artificial intelligence such as reasoning and learning; autonomous agents have the ability to modify their objective prior to execution; distributed agents could be executed on several distinct sys-

tems and multi-agent systems had the communication property and cooperation among several other agents to achieve common objectives [7; 21; 66; 77].

An example of agent application in UTC, is the use of multi-agent planning approach to implements a decentralized, schedule-driven traffic signal control [132]. Scalable Urban Traffic Control(SURTRAC) computes a schedule that optimizes the flow of currently approaching traffic at each intersection independently (and asynchronously). The resulting schedule is used to decide when to switch green phases and to indirectly influence the schedule of neighbouring intersection. This helps to provide real-time responsiveness to changing traffic conditions and coordinated signal network behaviour.

Despite numerous research on urban traffic control, there are still some challenges with UTC. The next section discusses some of the problems of UTC.

3.3 Urban Traffic Control Problem

Traffic congestion in urban roads often lead to increase in environmental pollution and could also leads to a strong degradation of the network infrastructure [110]. However, the need for efficient and robust integrated network management of urban traffic networks is still one of the problems of intelligent transport systems(ITS) [89]. Modelling and control of metropolitan traffic still remains a complex problem with lots of research efforts towards solving this problem [41].

Traffic control problem could originate from the design, installation, operation and control of signals. It could also originate from a fault in comprehensive maintenance function for a region's traffic signal network. However, a comprehensive and robust traffic control strategy would reduce the ripple effect of traffic problems on road users.

Urban Traffic Control should provide high-quality traffic signal control service to the district and the Highways Agency. Ideally, the following abilities would be incorporated into a road traffic control system.

Reduce travel time Car travel time and carbon emission will automatically be min-

imised when the flow of traffic are optimised irrespective of the road situation.

Give real-time information to Users Variable Massage Signs VMS, are used to influence the road users decisions while travelling on the road. This is done by the road monitoring officer to give out road situations and speed variations to road users. However, this could be done automatically with little or no human intervention by a UTC system. The VMS is part of the entire system and should give out information based on the processes and constraint faced by its internal component. This information can be accessed and use by road users to take decision while traveling on the road.

Give priority to some classes of vehicle This is the ability of a UTC system to give privileged to priority vehicles - such as ambulance, at road junctions.

Save energy Energy consumed by the traffic systems could be optimise at off peak time such as midnight till down when road usage in inner streets are close to zero. The ability of a road traffic system to change mode to sleep mode during this period will go a long way to save energy cost annually. This could be achieved when there is no vehicle detected on the street for a long period, with reference to the timing of the day.

Manage accident Accident and car brake down could impair traffic flow for a long time if the situation is not well managed. An intelligent UTC system can optimise this situation.

Store Traffic Knowledge An intelligence UTC should be able to store and retrieve the following:

- The knowledge of existing traffic situations
- Different traffic problem that was encountered and
- Processes and decision that was generated to solve those problems

Adaption and reuse of knowledge from past traffic situation Past knowledge should be use for accurate and precise future forecast through techniques such as Traffic Trend Mining.

3.4 Conclusion

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 [40; 79]. 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 that can reason about unforeseen situations in the road network and come up with plans (sequences of actions) achieving the desired traffic situation. According to Danko and Roozemond , there are three aspects where intelligent traffic control and management can improve current state of the art traffic controls (TC)systems: ability to learn and adapt its behaviour from previous situation; ability to communicate co-operate and turn signal plans and ability to exhibit proactive behaviour, this allows traffic system to plan ahead prior to execution [123]. This pro-active nature of traffic control is part of the contribution of this thesis.

3.5 Summary

This chapter gives a brief introduction to Urban Traffic Control, its components, application and problems. The next chapter explores the application of AI planning technology to problem of Urban Traffic Control(UTC).

Chapter 4

Applying AI Planning in an Urban Traffic Domain

This chapter explores the application of AI planning technology to the problem of Urban Traffic Control(UTC). This is accomplished by the creation of a declarative representation of a road network in a town center area in the United Kingdom. The task is to navigate vehicles effectively through a network of connected roads during an unforeseen situation. As part of this effort, we embed the knowledge of UTC structure into a planning domain and evaluate the possibility of reasoning with this knowledge and optimising traffic flow in situations where a given road within a network of roads becomes unavailable due to an unexpected situation such as road accidents.

4.1 The Road Traffic Domain Model

This section describes the design of a Road Traffic Domain Model (RTDM). From the automated planning perspective, we have to compromise between making RTDM realistic and the computational resources for solving RTDM planning problems. A very practical RTDM should be able to reason about continuous processes (e.g. cars are moving on the road continuously), uncertainty (e.g. a driver can decide its route or

alternative way); and unexpectable events could occur (e.g. traffic accidents). Continuous planning without reasoning about uncertainty is not well developed. On the other hand, classical planning (the simplest form of planning) is very well developed, however, not very suitable for RTDM since classical planning does not reason about time (i.e., Effects of actions are instantaneous). Hence, the reasonable compromise is using temporal planning that can in addition to classical planning reason about time (i.e. executing actions takes requires a time frame).

RTDM consists of two main parts – static and dynamic. The static part represents road network topology, i.e., roads, their capacity, length and junctions connecting the roads. The dynamic part stands for how many cars are on each road (and where) and whether the road is operational. The term 'operational' means that the road is available and accessible within the road network system. Clearly, the dynamic part is changing through the time as cars are moving through the road network.

4.1.1 Resources and Constraints

In every system design, there are resources available for execution. These resources come with associated limitations (constraints) which must be identified and optimised for effective implementation of the system. In UTC, we are dealing with various resources often limited by constraints. For example, we have road junctions, which can be understood as bottlenecks of traffic, where we must ensure that cars are not permitted to go through the junction in colliding directions. Also, each road had its capacity, i.e., the maximum number of cars on it. Situational awareness of the system can be sustained by various sensors that are placed on roads and intersections.

Each action has its duration in which is being executed. For instance, the duration of action for vehicle movement from a head to a tail of the road depends on the road's length and the speed of the vehicle.

1. Resources from the component of road traffic system which includes: road intersections, traffic lights, variable message signs, ramp metering and state switching at the intersection.

2. Resources from road sensors that include: status of roads and intersections, the capacity of the road and length of queues.

3. Constraints in this model could be further broken down into:

   Resource Constraint The execution of an activity, needs certain resources that must be optimised at every resulting state from, during and after every action within the domain.

   Temporal Constraint Every action have their duration that must be valid within any within the action in the execution framework. For instance, the duration of allowed action for vehicle movement is according to the road's queue length in relation to the road capacity of neighbouring routes.

   Conditions and Effects Constraints Action duration specifies numeric time frame at which the condition and effect become true. For instance, the action "RELEASE" would make it possible for a vehicle to move at the start and disable the movement after a period of time.

4. *System functions* Ability to switch between allowing and stopping of vehicles on the road, while maintaining good traffic flow and minimising travel times. They should satisfy all the preconditions necessary to change the status of the road and the after effect based on the available resources.

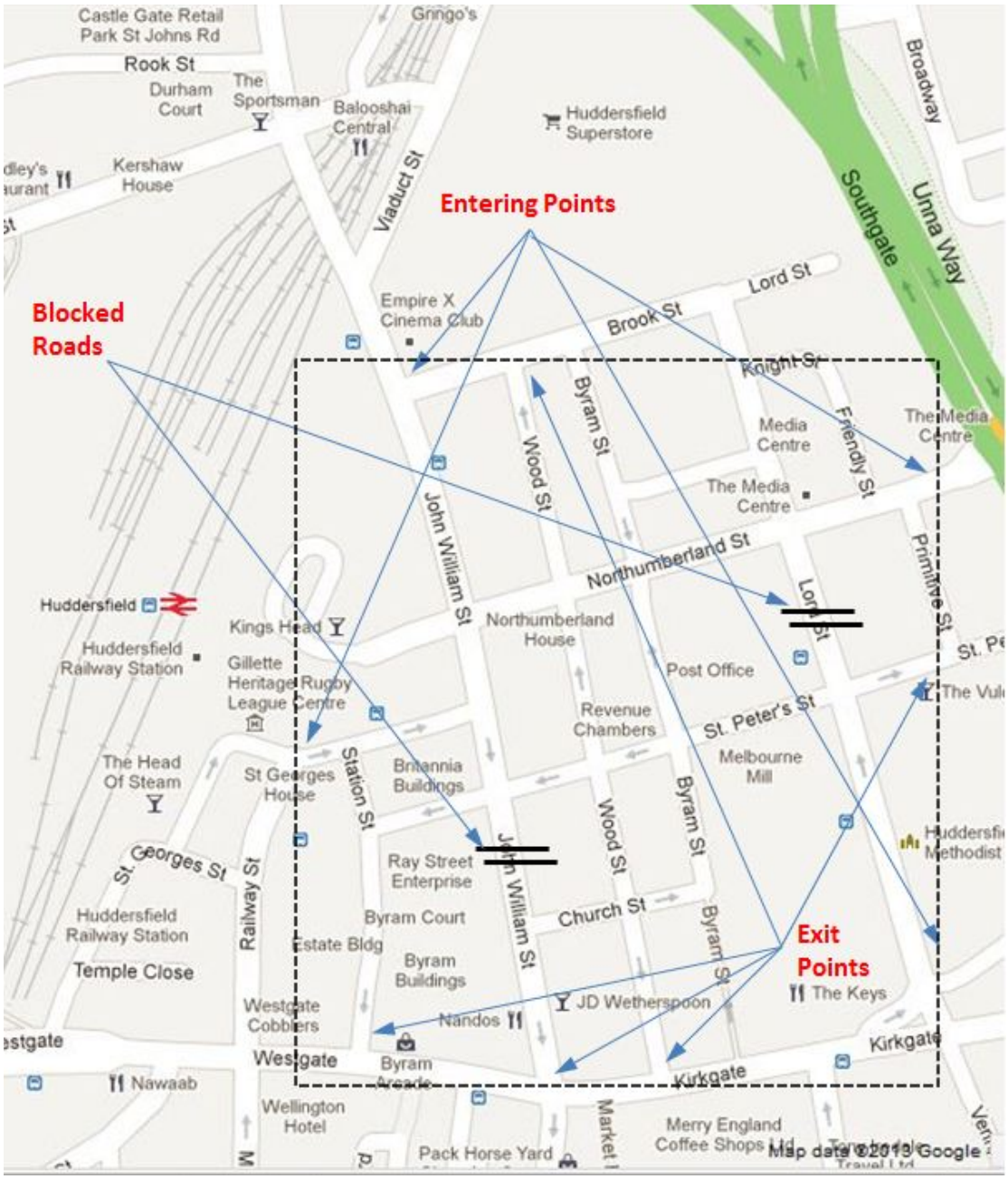

Figure 4.1: Map showing the network entry and exit points and the blocked roads in a part of town center of Hudderfield, West Yorkshire, United Kingdom. It is used for our empirical analysis.

4.1.2 RTDM Specification

This section shows the RTDM specification(originally derived in [79]). A Road Network can be represented by a directed graph, where edges stand for roads and vertices stand for either junctions, entry or exit points. Entry points are places where cars enter the network while exit points are places where cars exit the network. At junctions, we must take into account the fact that, in some directions, cars cannot go through the junction simultaneously. Hence, we have to specify sets of mutually exclusive (mutex) directions by which cars cannot pass through the junction simultaneously. Every road has its length (determining how long it takes for a car to through it) and capacity (i.e. a maximum number of cars it can serve). The network model is enhanced by considering time intervals when a road is not operational (e.g. closed due to an accident).

Let (V,E) be a directed graph such that $\forall v \in V : (indeg(v) = 0 \Rightarrow outdeg(v) = 1) \wedge (outdeg(v) = 0 \Rightarrow indeg(v) = 1)$. Edges in E represent one-way (or one direction of two-way) *roads*. A vertex $v \in V$ represents:

- *entry point* if $indeg(v) = 0$
- *exit point* if $outdeg(v) = 0$
- *junction* otherwise

Let $\mathcal{M} = \{M_1, \ldots, M_n\}$ be a set of sets of *conflicting ways* such that M_i is defined as a set of triples in form (e_{i_x}, v_i, e_{i_y}) such that a junction v_i is a tail of e_{i_x} and a head of e_{i_y}.
Let $C : E \to \mathbb{N}$ be a function representing *road capacity* and $l : E \to \mathbb{R}$ be a function representing *road length*.
Let O be a mapping from edges (E) to sets of time intervals representing *operationality* of the road. Then $\mathcal{N} = \langle V, E, \mathcal{M}, C, l, O \rangle$ is a *Road Network*.

Given the definition of the Road Network, which gives us constraints referring to a static part of the environment, we have to specify a dynamic part of the environment which is related to traffic. We define a function $use : E \times T \rightarrow \mathbb{N}_0$ which refers to a number of cars on the road in a given time-stamp. Clearly, it must hold that $\forall t \in T, r \in E : use(r,t) \leq C(r)$. It is also necessary to distinguish whether cars are on the head or the tail of the road. For this purpose we define functions *head* and *tail* such that $head : E \times T \rightarrow \mathbb{N}_0$ and $tail : E \times T \rightarrow \mathbb{N}_0$.

An RTDM Planning Problem addresses the problem of effective navigation of cars through a given Road Network from entry points to exit points. To achieve this, we have to take two types of decisions, i.e., which way cars should take and when traffic lights should be set to green or red. Initially, it is given the number of cars at each entry point and frequency of their releasing. This can be represented by a set time-stamps in which the entry points are 'opened'. The goal situation is determined by numbers of cars in exit points while minimising 'makespan', the time needed to navigate all the cars through the road network. In our model we consider four planning operators which are defined as follows. Note that Δt stands for operator's execution time which might differ for different operators' instances (e.g driving through different roads may take different amount of time).

- In a given time-stamp t, an operator $release\text{-}cars(r,n,t)$ releases n cars to the head of the road $r \in E$ if r is an outgoing edge from a given entry point and $\forall t' \in \langle t; t + \Delta t) : C(r) \geq use(r,t') + n$ and r is operational in t'. The effect of this operator is that $use(r,t + \Delta t) = use(r,t) + n$ and $head(r,t + \Delta t) = head(r,t) + n$.

- An operator *drive-through-junction*(r_1, v, r_2, n, t) navigates n cars from the road $r_1 \in E$ through the junction $v \in V$ to the road $r_2 \in E$ if $tail(r_1, t) \geq n$, $\forall t' \in \langle t; t + \Delta t) : C(r_2) \geq use(r_2, t') + n$, r_2 is operational in t' and no instance of operator *drive-through-junction*(r_x, v, r_y, n_x, t_x) is executed such that $t_x \in \langle t, t + \Delta t \rangle$ and $\nexists M \in \mathcal{M} : \{(r_1, v, r_2), (r_x, v, r_y)\} \subseteq M$. The effect of this operator is that $tail(r_1, t) = tail(r_1, t) - n$, $head(r_2, t + \Delta t) = head(r_2, t) + n$, $use(r_1, t) = use(r_1, t) - n$ and $use(r_2, t) = use(r_2, t) + n$.

- An operator *drive*(r, n, t) moves n cars from a head of r to its tail in a time-stamp t. That is if $head(r, t) \geq n$, then $head(r, t + \Delta t) = head(r, t) - n$ and $tail(r, t + \Delta t) = tail(r, t) + n$.

- An operator *exit-cars*(r, n, t) allows n cars to leave the network in a given exit point (r is an incoming edge to the exit point) in a time-stamp t. If $tail(r, t) \geq n$, then $tail(r, t + \Delta t) = tail(r, t) - n$.

4.1.3 Modeling RTDM in PDDL

Modeling an RTDM problem in PDDL(explained in Chapter 2.0) holds several challenges. The most critical of these is that in real-world drivers have true agency and are therefore not strictly under our control. Thus, in a real world scenario, we propose the use of variable message signs(VMS) with strict traffic signal head control. This VMS should be connected to the planner so that the re-direction and traffic calming and metering can be uploaded and communicated from the planner to road users in real time. The control plan should also be used to switch signal heads(an assembly of one or more signal faces) in other to enable or disable the flow of traffic for desired optimal operation.

```
(:durative-action DRIVE
 :parameters (?r - road ?n - num)
 :duration (= ?duration (length ?r))
 :condition
  (and
    (at start (>= (head ?r) (val ?n)))
    (over all (operational ?r)
  )
 :effect
  (and
   (at start (decrease (head ?r) (val ?n)))
   (at end (increase (tail ?r) (val ?n)))
  )
)
```

Figure 4.2: The PDDL encodings of the *drive* planning operator.

This in turns influence and control the decision of drivers to a large extent. The flow of traffic is, therefore, best modelled as a complex hybrid process. However, this process is difficult to define and would lead to a problem too challenging for contemporary temporal planners. The next chapter explained how we resolve this problem using our continuous planner.

For our purpose, we have used PDDL 2.1 [57] since it encapsulates features needed for representing temporal planning domains and problems. The environment of RTDM is modelled by using predicates or fluents. Predicates refer to relations between objects, for example, if a predicate (operational r1) is present in some state, then in this state the road r1 is operational, otherwise not. Fluents can operate with more (numerical) values, for instance, a fluent (use r1) represents a current use of the road r1 which is in a range $\langle 0, C(r1) \rangle$.

Figure 4.2 depicts the planning operator *drive* (in PDDL 2.1 planning operators are called 'durative actions'). Time-stamps in which the operator can be executed are not explicitly modelled as operator's parameter in PDDL, only duration of operator's execution is defined :duration (= ?duration (length ?r)).

Time-stamps are, therefore, modelled relatively. Given the notation from the previous subsection, i.e., the operator is executed in a time-stamp t and the duration of its execution is Δt, *at start* refers to t, *at end* refers to $t + \Delta$ and *over all* refers to the interval $\langle t, t + \Delta t \rangle$ (note that *over all* can be only used for preconditions). To keep the consistency of the environment we cannot apply effects at the same time the precondition is being checked (e.g. apply (at start (decrease (head ?r) (val ?n))) at the same time when (at start (>= (head ?r) (val ?n))) is being checked). This issue is handled by applying the effect just after the precondition has been checked, so for a very small ε the effects are applied in $t + \varepsilon$ (*at start*) or in $t + \Delta t + \varepsilon$ (*at end*). Another issue in PDDL 2.1 is impossibility to have a numeric parameter (all the parameters must be object types). This issue can be handled by introducing a special object type num and a fluent (val ?n) which maps a numerical values to num objects.

In all considered models, the goal of the planning problem is to route a certain number of vehicles through the network to the exit points, while minimising the makespan of the plan irrespective of any disruption in the network of connected roads.

A typical RTDM problem requires a method of releasing cars at the entry points at given (periodic) time-stamps. Releasing cars can be understood as an event. However, events are not supported in PDDL 2.1. Releasing cars by the operator *release-cars* is executed at any time when its precondition is met. This issue can be overcome by using timed initial literals which is a feature supported by PDDL 2.1. We introduce a fluent (ready ?x) for each entry point, representing the number of cars that are ready to enter the road network.

Therefore, several statements of the form:

```
(at  0 (= (ready a) 5))
(at 10 (= (ready a) 5))
       ....
```

can be added to the initial state.

We believe that this is an attractive approach as it ensures the flow of traffic into the system in a more realistic way rather than simply adding all vehicles at time 0. Also, we might use timed initial literals for defining temporary road blockages.

Unfortunately, the use of fluent assignments in timed initial literals is not supported in PDDL (or at least by the planners we experimented with), and so another approach was necessary. We use the unary predicate *ready* to denote that an entry point is ready to release cars with the duration of a *RELEASE-CARS* action, this is deleted to ensure that only a single instance of the ground action can be executed at any one time. At the end of the action, the entry point is set to ready again. The duration is set to the amount of time successive waves of vehicles can enter the network. Therefore, several statements of the form:

```
. . . .
(ready a)
   (at 0.1 (not (ready a)))
   (at 10.0 (ready a))
   (at 10.1 (not (ready a)))
   (at 20.0 (ready a))
    . . .
```

A weakness of this approach is that a planner is free to schedule *RELEASE-CARS* actions as far apart as it chooses. In this respect, the domain does not adequately model the fact that the vehicles should enter the network at regular intervals, regardless of when the planner chooses to release them. We do not know of a remedy to this situation other than the use of timed initial literals; and as previously discussed, this proves impractical with current planners. Pragmatically this problem is reduced by the fact that for the makespan to be optimized it is sensible to release the cars as early as possible.

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 4.1: 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.

4.2 Evaluation

For evaluation purposes, we selected known planning engines Crikey [32], Optic [12] and LPG-td [64] which are capable of handling PDDL 2.1 features (including timed initial literals). LPG-td, however, is not very useful for our problem for 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.

The experimental setting consists of several 'specific' problems that are defined on the top of the road network depicted in Figure 4.1. Uncertainty is not considered in our experiments. 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 in the south, north and east exit points (individual vehicles are not distinguished). Problem 3 has the same settings as 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.

4.3 Result

Table 8.1 shows the results of the preliminary experimental settings using the Crikey and Optic planners. We can observe 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 that could lead to road blockage. Also, given the fact that in a real-world environment, cars are entering the road network continuously, which we have not been able to model reasonably (it is discussed later), good quality of plans (regarding makespan) can somehow ensure that the traffic flow remains continuous and roads will not become congested.

A plan depicted in Appendix 3, a solution of Problem 1 (no blockage), generated by the Optic planner, shows some interesting aspects. Firstly, despite going from the same entry point to the same exit point cars are split during 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 much 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. Such a plan, when executed, allows one car to go through the junction, and then the other cars have to wait until that car drives through to the connected road. This approach is quite counter-intuitive and consequently to this plans are not optimal.

Another observation is that both Crikey and Optic do not handle timed-initial literals well. For example, if a road is blocked only for a given time interval, then the planners conclude that there does not exist a solution despite the fact that a solution exists for the same problem whenever it is blocked for the entire planning duration.

4.4 Challenges

The PDDL domain model does not allow the explicit modelling of complex resource consumptions that are specific to UTC environment. The planners also loop endlessness without generating a plan, whenever an attempt is made to specify explicitly most of the resource constraint in the domain description. Also, the planners fail to produce a plan whenever the numeric fluents within the domain description are increased to higher values of real world traffic scenarios. For example, changing $(q = 2, 5, 10, ...)$ to $(q = 20, 50, 300, ...)$ where $q = queuelenght$ of cars on roads in meters.

These setbacks show that using state-of-the-art domain-independent planning engines does not lead to optimal solutions even in quite simple cases. Also, planners' performance 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.

4.5 Conclusion

In summary, embedding planning component into the UTC might be beneficial since it enables (centralised) reasoning in a given area which, for instance, can more easily overcome non-standard situation (e.g. road blockage after an accident). Our results show 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.

4.6 Summary

This chapter explores the application of AI planning technology to problem of Urban Traffic Control(UTC) system. This is accomplished by the creation of a declarative representation of a road network in a town center area in the United Kingdom. The task is to navigate cars effectively through a network of connected roads during an unforeseen situation. As part of this effort, we embed the knowledge of UTC structure into a planning domain and evaluate the possibility of reasoning with this knowledge and optimising traffic flow in situations where a given road within a network of roads becomes unavailable due to an unexpected situation such as road accidents.

Chapter 5

Introduction to Model Predictive Control

This chapter introduces model predictive control as a branch of control engineering. It explains the strategy and approach of MPC in controlling and stabilising process control within the field of Engineering. This chapter ends by comparing the similarities and difference between MPC and AI planning to make a judgement call for a hybrid solution that incorporate both technologies to solve problems involving both discrete and continuous state variables.

5.1 Introduction to Model Predictive Control

Control engineering is a field of knowledge within the engineering discipline; that applies control theory to design and implement systems with desired behaviors. Predictive controls are a branch of controls engineering that are used in adapting and forecasting the future trend of control processes in other to manipulates it inputs for a desirable result in a future time. There exist different types of predictive controls, for example, the Receding Horizon Predictive Control(RHC); the Generalised Predictive Control (GPC) and the Model Predictive Controls (MPC).

MPC has attracted notable attention in the control of dynamic systems and has gained a significant role in the control practice [140]. MPC was developed in the industrial area has an alternative algorithm control to the conventional Proportional Integrate Derivative [11](PID) controls. MPC formulation integrates optimal control, stochastic control, control of processes with dead time, multi-variable control and future references when available [20].

There are various MPC algorithms, they differ in the way they represent the model of the process, disturbance and the cost function to be minimised. Its algorithm has been constantly improved and refined to increase its robustness for real-time processes [1; 50; 135; 140]. Some of the well-known algorithms include: Dynamic Matrix Control; Model Algorithmic Control; Predictive Functional Control; Extended Prediction Self-Adaptive Control; Extended Horizon Adaptive Control and General Predictive Control.

It has been implemented in varieties of application processes ranging from production planning [92]; industrial production [136] and supply chain [117];Intelligent Transport Systems [90]; agriculture [6] and robot manipulation in path planning [38; 45; 49; 82]. MPC is suitable for most control problems, however it displays its main strength when applied to problem involving:

- A large number of manipulated and controlled variables
- Constraints imposed on both manipulated and controlled variables
- Change in control objectives or system failure(sensors, actuators)
- Time delay

5.1.1 MPC approach

The mathematical model of controlled process, as well as the assumed disturbances that might occur during its operation, is built based on the past experience of operation and past data from similar operations within the same system. A cost function is derived

from the available resources and constraints that need to be optimised for the entire duration of the process. The system uses the pre-defined model as a guide to maximise the cost function, given: a set of varying input parameters, output parameters and the dynamic changes in the state of the environment. The system plan over a period of time in the future, which is known as the horizon. The generated plan is applied to the system to control the process by changing it current state to a desirable state for a given period of time. The new state is sampled again. It re-plans for another horizon taking the present state from the feedback loop as well as all the system constraints into consideration. This approach of planning is called, ”receding horizon”. This planning and re-planning approach makes MPC robust and able to keep a control process in a desirable state for an extended period of time. It also allows it to function in a partially observable environment, because of its ability to sample dynamic environment at every sampling time during a re-plan.

5.1.2 MPC Concepts and Terminologies

MPC is used to predict the future behaviour of processes or output of a system over a period of time in the future. This is achieved by computing the future input parameters at each step while minimising a cost function under disparity constraints on the manipulated controls and controlled variable. MPC applies only the first set of control variables on the controlled system and repeats the previous step with new measured parameters [140].

Basically, an MPC strategy entails three main steps:

Past Make use of past data to model a system

Future Predict the impact of future control event with the help of the model

Present Select and perform the control action that is expected to yield the best long-term outcome.

MPC Elements MPC is basically composed of three elements

- The Predictive Model
- The Objective function
- The Control Law

The Predictive Model

The prediction Model comprises of a model of the process under consideration, a model of the possible disturbance that could affect the process, the Free and Forced Response or both. MPC model formulations involve the combination of different methodologies with the use of varieties of models for the specific control process. Some of the approaches that can be employed during modeling of a control process include:

- Impulse response
- Step response
- Transfer function
- State space
- Others

The Truncated Impulse Response Model measures the output whenever the process is excited with an impulse from the input. It allows complex dynamics such as non-minimal phase to be described, and the identification process simplified with no need for prior information about the process. For instance, given an impulse response whose output is related to the input by: $y(t) = \sum_{i=1}^{\infty} h_i u(t-i)$ where h_i is the sampled output when the process is excited by an impulse.

Then the prediction might be: $\hat{y}(t+k|t) = \sum_{i=1}^{N} h_i u(t+k-i|t)$

Closely related to impulse response is the Step response model which measures the output when input is stepped. The step response share both advantages and disadvantages with impulse response. For a given stable system, the truncated response could be given by : $y(t) = y_0 + \sum_{i=1}^{\infty} g_i \Delta u(t-i)$ where g_i are the sampled output value for the stepped input and Δu is the change in input values written as $\Delta u = u(t) - u(t-1)$

The prediction could be : $\hat{y}(t+k|t) = \sum_{i=1}^{N} g_i \Delta u(t+k-i|t)$

The Transfer Function model, on the other hand, requires only a small parameter and is valid for all kind of processes; a prior knowledge of the process is essential in this case though it has an advantage of the fact that it only needs a few parameters.

The State-Space Model uses current information about the state variable representation at a given time to predict ahead the future control input and output of the system. It is also utilised in some formulations for describing multivariable processes.

For instance, a single-input-single-output system can be described by:

$x(k+1) = Ax(k) + Bu(k)$ and $y(k) = Cx(k)$ where u is the manipulated variable or input variable; y is the process output and x is the state variable; A, B are input matrix of the system and C is the output matrix of the system with the assumption that there is no disturbance in the system.

The prediction of the system output trajectory could be :

$\vec{y} = \bar{P}x(k) + \bar{H}\vec{u}$

where

$\vec{y} = [y(k)^T y(k+1)^T ... y(k+N-1)^T]^T$ and

$\vec{u} = [u(k)^T u(k+1)^T ... u(k+N-1)^T]^T$

MPC Models can also be represented by a non-linear model, neural network and fuzzy logic in some applications.

The Objective Function

The objective function is the generation of a criterion for an optimiser to predict the next future input to a control system. It comprises of the constraints of the system; the system parameter and the reference trajectory(goal condition). The objective function generates the required cost function for obtaining the control law.

The Control Law

An MPC control law is a control plan that needs to be executed by the controlled system to keep the system in a desirable state. The process of obtaining the control law is achieved by differencing the actual performance to the expected performance(using the objective function) to obtain the future errors. The optimizer calculates next sequence of input actions taking into consideration the future tracking error, the cost function and the constraints. The optimiser provides the control action which it an important factor in any MPC strategy.

5.1.3 MPC Strategy and Structure

A process model is used to predict the future output for a determined prediction horizon L at each instance of time t. This predicted output $\hat{y}(t+k|t)$ indicate the value of the variable at the instance $t+k$ calculated at instance of time t for $k = 1,...,L$. The predicted output depends on the known values up to instance of time t which includes the previous input and output values and on the future control input values $u(t+k|t)$, $k = 0$, . . ., $L-1$. The process is kept as close as possible to the reference trajectory w. The reference trajectory is usually a set point or an approximation of a set point that needs to be maintained for an optimal performance of the process. The reference trajectory is a goal condition that has to be achieved and maintained over a period of time.

This is achieved by using some defined criterion to create a future control plan that helps to keep the process closed to the fixed reference trajectory. The control criterion is usually the error between the predicted output plan and the reference trajectory. This error is included in the objective function and used as part of the input to the optimiser to generate future input $u(t+k|t)$ to the model. This loop sequence of activities is repeated within the control loop of Figure 5.1 while Figure 5.2 illustrates the structure for the implementation of this strategy.

For instance, a general model could be derived from a sample process model:

$y(t) + a_1 y(t-k) + ... + a_n y(t-nk) = b_1 u(t-k) + b_2 u(t-2k) + ... + b_n(t-nk)$

Where u is the input to controller, y is process output, and k is the time interval. This sample general equation relates previous process output $y(t-k)$ and previous controller input $u(t-k)$. This equation is often derived from knowledge of specific systems in control engineering. It can sometime be created from past experimental data. At time t, total previous behavior y_p becomes: $y_p = f(y(t), y(t-k), ..., u(t-k), u(t-2k), ...)$

While the future output process y_f can be predicted using present control $u(t)$ and future input control $u(t+k)$: $y_f = f(u(t), u(t-h), ..., u(t+Nk))$

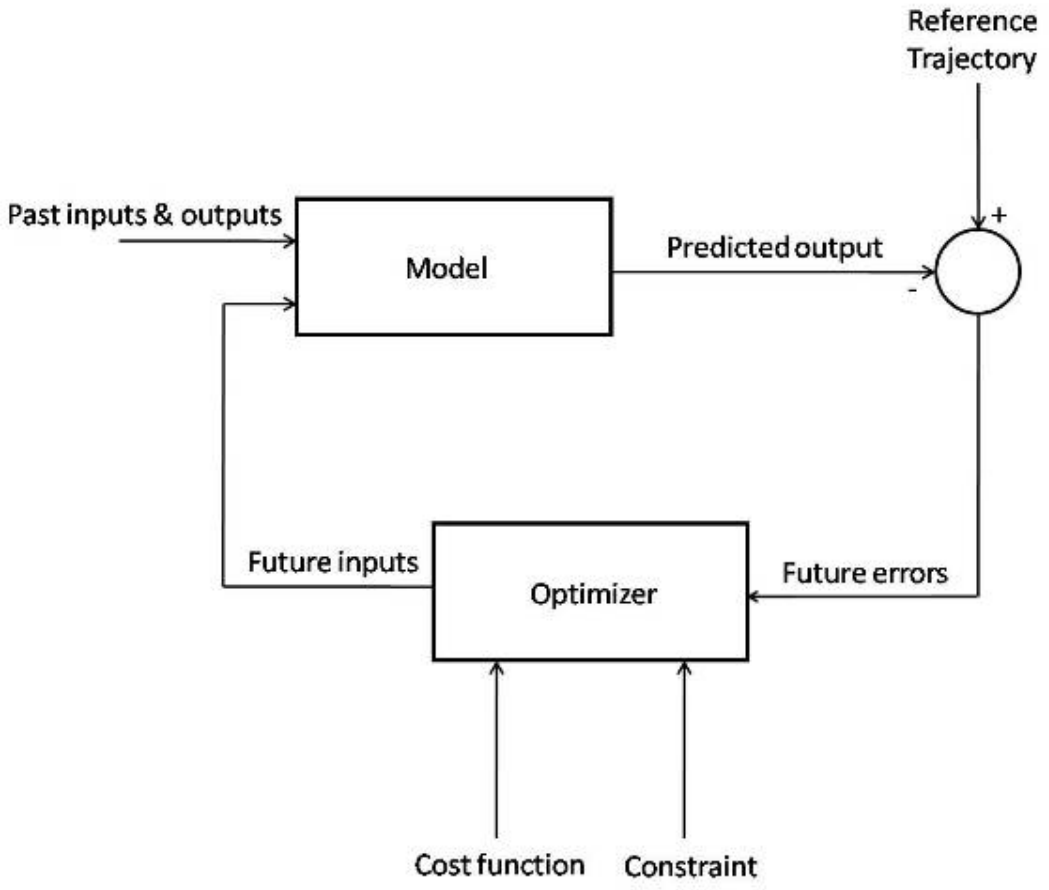

Figure 5.1: Model Predictive Control Feedback Loop

Both y_f and y_p could be created from experimental data, but are usually derived from explicit equations related to a given system. Any irregularities from the expected system behaviour y_d produce an error function $e(t)$ which could be computed as: $e(t) = (y(t) - y_d(t))$ for a change in control actions $\Delta u(t) = u(t) - u(t-k)$).

The objective function J to be minimized could be in the form of: $J(u(t), u(t-k), ..., u(t+kh)) = \sum_{k=1}^{t+N} e(t+hk)^2 + \rho(\Delta u(t+(h-1)k))^2$

The first set of the control inputs that minimize the above objective function can be applied to the system by the controller over the time interval and the process is repeated.

It should be noted that the control input function F could also be determined implicitly by optimization. $u(t) = F(y(t), y(t-k), ...y(t-nk), u(t-k), y(t-2k), ..., u(t-nk))$

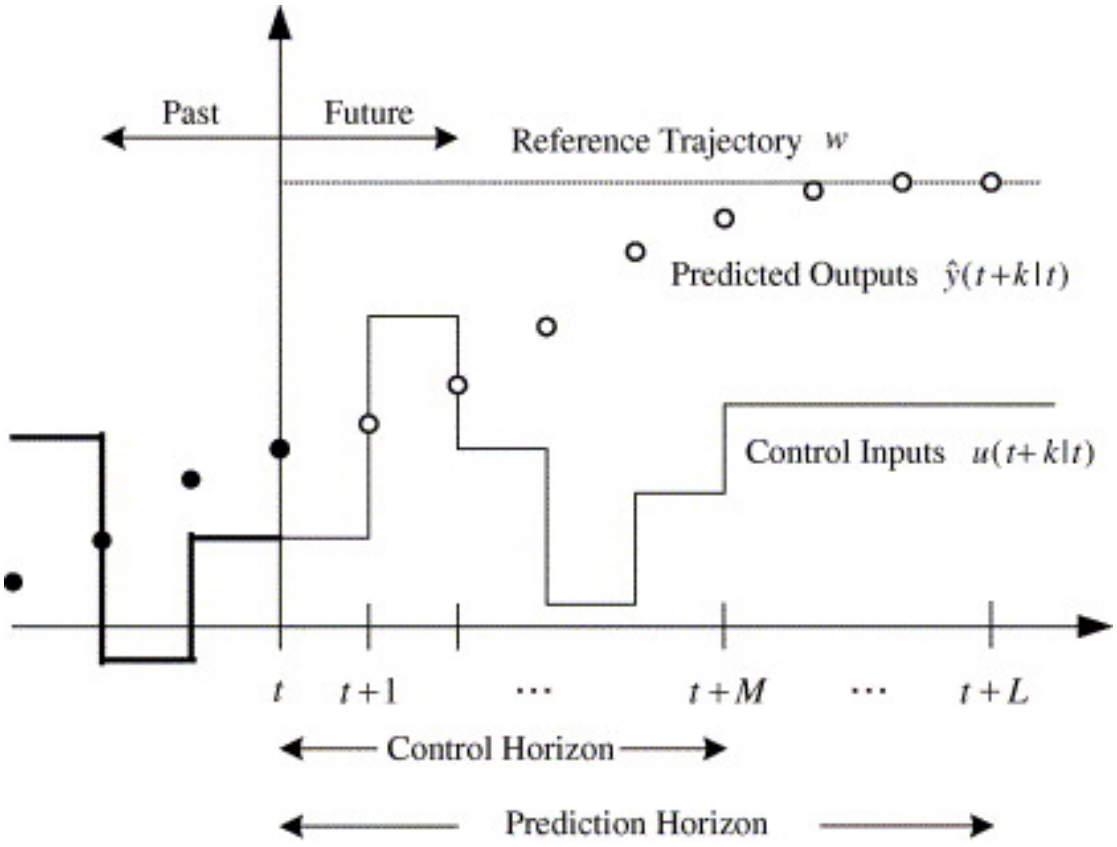

Figure 5.2: Graph to Depict the Structure of an MPC

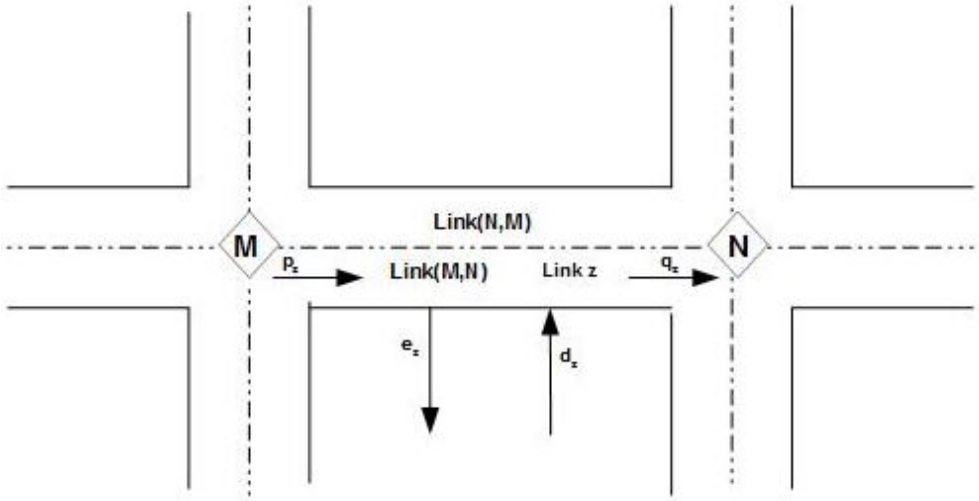

Figure 5.3: Representation of urban roads using links and junctions

5.2 Sample MPC Model Analysis

Traffic flow models are of three distinguished types: macroscopic model; microscopic model and mesoscopic model. A detailed overview of existing traffic models could be found in [75]. Macroscopic models is considered in this analysis through the use of aggregated variables to describe traffic flows [67]. Macroscopic models are suited well for online control such that, the prediction will be implemented online in a traffic optimization operation. A store-and-forward traffic flow model is used to formulate a state-base MPC model, this enables a mathematical description of the traffic flow, using state space method. store-and-forward traffic flow model was developed in 1963 by of Gazis and Potts with the hope of getting a balanced trade-off between control accuracy and computational efforts. The next subsections show the store-and-forward flow model that depict the urban traffic control model.

5.2.1 Store-and-forward modeling

Roads networks can be described as sets of links $z \in Z$ and junctions $j \in J$ as shown in Fig 5.3. There are sets of incoming links I_j and outgoing links O_J in each signalized

junction j. Fig 5.3 depicts a sample urban roads with two neighboring junctions M and N ,where $z \in I_N$ and and $z \in O_M$. The remaining essential variables are:

$x_z(k)$ number of vehicles in link z, this refers to the length of queue at step k ,which implies the state variable

$g_{j,i}$ the green time of stage i at junction j , this is the control input

S_z saturation flow of link Z

$t_{w,z}$ turning rate towards link Z from the links w that enter junction M

C_j cycle time of junction j

T discrete time step, this is the control interval

v_z the set of stages where link z has right of way

k discrete time index, $k = 0, 1, 2...$

j the junction identifier

i the stage identifier

Assuming that the cycle times C_j for all junctions $j \in J$ are equal and fixed, namely $C_j = C$. The dynamics of link z can therefore be represented by the conservation equation:

$$x_z(k+1) = x_z(k) + T[(1-\tau_z) \sum_{w \in I_M} \frac{S_w \sum_{i \in v_w} g_{M,i}(k)}{C} - \frac{S_z \sum_{i \in v_z} g_{N,j}(k)}{C}] \tag{5.1}$$

Saturation flow S_z gives the outflow capacity of each link $z \in Z$ during its green time. Assuming the exit rates τ_z are known. S_z is also assumed to be known and constant, this could be calculated by another approach or using a standard value. Turning rates $t_{w,z}$ of $w \in I_j$ and $z \in O_j$, are also set using a statistical value or estimated in real-time.

Assuming $T = C$ while replacing the second and third term with some simplified variables, equation 5.1 can be described as:

$$x_z(k+1) = x_z(k) + T[p_z(k) - q_z(k) + d_z(k) - e_z(k)] \tag{5.2}$$

where $p_z(k)$ are the inflow to link z and $q_z(k)$ are the outflow from link z, in the sample time $[kT, (k+l)T]$; $d_z(k)$ and $e_z(k)$ are the demand and the exit flow in the link z, respectively. The exit flow $e_z(k)$ can be estimated by $s_z(k) = \tau_z p_z(k)$. The outflow can be represented by:

$$q_z(k) = \frac{S_z \sum_{i \in v_z} g_{N,i}(k)}{C} \tag{5.3}$$

The modeled flow does not consider red-green switching in a cycle to reduce computational efforts. However, the flow is an average of real flow for each period.

Given a linear scalar equation for describing a given link as shown in equation 5.1. Organising all interconnected conservation equations for each link in a state space form, a state space model defining a whole traffic network would be given by:

$$x(k+1) = x(k) + Bg(k) + d(k) \tag{5.4}$$

where $x(k)$ is the state vector representing numbers of vehicles in each link(queue lenght); **B** is a constant coefficient matrix of proper dimensions representing the network characteristics, such as topology and turning rates; $g(k)$ is the control vector representing all green time settings and $d(k)$ is the disturbance within the network.

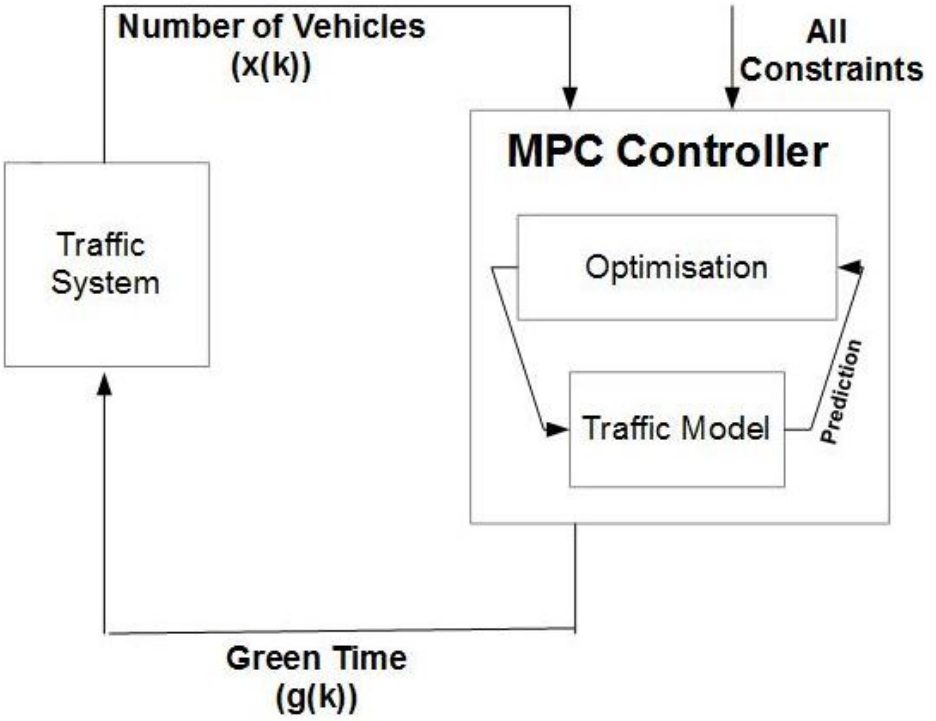

Figure 5.4: Applying MPC to UTC structure

5.2.2 MPC Constraints Example on UTC Model

Given a UTC traffic model, there are some constraints that have to be considered. The constraints are formulated from the store-and-forward model discussed in the previous section

5.2.3 Non-negative control constraints

The number of vehicles that moves through link z at time k is a non-negative number.Given Thus, the green duration for a direction at an intersection is between a traffic light cycle:

$$g_{j,i} \geq g_{j,min}, \forall i \in J \tag{5.5}$$

5.2.4 Traffic Light Cycle Constraints

The constraint of control input which is the green time holds at junction j in stage i : The queues on a link are subjected to the length of a link between two junctions:

$$\sum_{i=I}^{N_j} g_{j,i}(k) \leq C - L_j, \forall j \in J \tag{5.6}$$

where N_j is the number of stages at junction j . L_j is the fixed lost time at junction j.

5.2.5 Green Duration Constraints

The upper and lower bounds of green time is given by:

$$g_{j,min} \leq g_{j,i} \leq g_{j,max}, \forall j \in J \tag{5.7}$$

where $g_{j,min}$ and $g_{j,max}$, represents the minimum and maximum permissible time at junction j respectively, to set enough green time for other phases at the junction, such as the pedestrian phase.

5.2.6 Flow Conflict Constraints

Two links may collide at an intersection. In this case, at most one link may be active at a time.

5.2.7 Non Negative Queue Constraints

The queues on a link are subjected to the length of a link between two junctions:

$$0 \leq x_z \leq x_{z,max}, \forall z \in Z \tag{5.8}$$

Where $x_{z,max}$ is the maximum admissible numbers of the vehicles in link z. This limitation helps to avoid over saturation of a link in rush hours. It also makes sure that

the value of a queue length on a road is non-negative during the computation of control input.

5.2.8 Capacity Constraints

The capacity of a link must not be exceeded. Thus, the number of leaving vehicles of any link will be limited by the state and capacity of the downstream link.

5.2.9 The Objective Function

The objective of this MPC formulation is to reduce the number of vehicles waiting in line at a junction. This is evaluated in Chapter 8. as the total time it requires for our planner to exit the vehicles waiting at individual connected junctions in the the network. Thus, to minimise the number of vehicles in queues, a quadratic costs function satisfying 5.4, 5.6 and 5.7 is given by 5.9, with the aim of minimising queues and green times at a junction.

$$J = \sum_{k=1}^{N_p} (\| x(k) \|_Q^2 + \| g(k) \|_R^2) \tag{5.9}$$

5.3 Comparison between MPC and AI Planning

The ability to reason with a model makes MPC an attractive technique from the point of view of AI planning. The MPC control structures shares similarities with AI planning strategies. Some of the similarities include:

1. Given a domain of interest with facts and description of the environment and problems within that domain, a model could be defined as a construction of a symbolic system that has inference and rules that represents the domain of interest. Both MPC and planning require modeling of the system under consideration.

2. State space MPC(detailed above) is also similar to State Space planning approach (detailed in Chapter 2.)

3. MPC applications are mostly implemented in continuous processes like chemical process. AI planning is now being introduced in continuous domains such as chemical processes with the aim of modelling continuous processes and events.

4. MPC continuously generates a sequence of control plan to keep processes operation in a desirable state. AI Planning with continuous state variables also aims to generate control sequences that can keep a process in a desirable state.

5. MPC produces control plans by gradually work towards maintaining a reference trajectory(explained above). AI planning, on the other hand, generates series of state transitions that gradually work towards a state satisfying the goal conditions (similar to reference trajectories).

Despite all the similarities, MPC and AI planning also have some differences which include:

1. MPC makes use of a functional mathematical model to represent numeric values of the system in consideration while planning uses logical models augmented with numeric and function representations.

2. In most cases, the input variable to an MPC is in the form of complex numeric values from sensors such as temperature, flow rate and pressure. In Planning, planners take set input of input values in the form of a text file such as problem file containing declarative statements about the (initial) state and goal.

3. MPC generates control laws by converting the problem into an optimisation problem (QP/LP) that can be solved by a solver while minimising a cost function. AI planning, on the other hand, converts the planning problem primarily into a search problem (with few exceptions) using some heuristic function to cope with the combinatorial explosion within the search space.

4. MPC takes care of Multiple Input and Multiple Output (MIMO) constraints. This features making it very suitable for multimodal domains such as Urban Traffic Control (UTC). However, some existing domain independent planner would fail in multimodal domains due to inability to handle complex constraints.

5. MPC has the ability to predict the state of the future output of a system using receding horizon in a feedback loop that compensates for dynamic changes in the system. While classical AI planning predicts future states, generating necessary steps that could lead to a desired future goal condition without taking the dynamics of the environment into consideration(offline planning). Thus, some AI planning implementations are offline while MPC implementation must be online in order to have real-time control of the continuous process.

5.4 Conclusion

Exploiting the relationship and building on the synthesis of MPC and AI planning techniques to solve problems involving both discrete and continuous state variables, is the heart of this research work. We consider an Urban Traffic Control (UTC), because its model consists of both discrete(e.g., traffic light switch) and continuous(e.g., traffic flow rate) state variables. The next chapter explains our perspective of UTC and our self-management architect that aimed at using both AI planning and MPC approach in creating self-management properties in UTC. Chapter 6. describes the hybrid planner that is built from knowledge of these two technologies.

5.5 Summary

This chapter introduces model predictive control as a branch of control engineering. It explains the strategy and approach of MPC in controlling and stabilising process control with the field of Engineering. This chapter ends by comparing the similarities and difference between MPC and AI planning to make a judgement call for a hybrid solution that incorporate both technologies to solve problems involving both discrete and continuous state variables. The next chapter introduces the hybrid environment of interest that is used for the evaluation of this research work - Urban Traffic Control.

Chapter 6

A Self-Management System for Urban Traffic Control

This chapter describes the design of a generic architecture which is aimed at using both AI planning and MPC approach in creating self-management properties in UTC. Our architecture consists of several blocks. This chapter describes each of the blocks in the context of an Urban Traffic Control Domain.

6.1 Introduction

The need for planning and execution frameworks has increased interest in designing and developing system architectures that use state-of-the-art plan generation techniques, plan execution, monitoring and recovery to address complex tasks in real-world environments [88; 104; 105]. An example of such an architecture is T-Rex (Teleo-Reactive EXecutive): a goal-oriented system architecture with embedded automated planning for onboard planning and execution for autonomous underwater vehicles to enhance ocean science [101; 120]. Another 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 experi-

ences [78].

A similar research to our architecture in the field of UTC is Urban Traffic Control system using self-organization by [127]. This work is inspired by a technique exploited by social insects to coordinate themselves and specialize their behaviour, without any centralized coordination or explicit communication. It uses a distributed approach at each intersection. Each intersection has a controller that controls local traffic by executing simple local reactive rule-based policies. Every intersection controller selects on its own which policy to use in response to stimuli perceived from the environment through the use of sensors. This approach implies that each local control is indirectly being influenced by the decisions of other adjacent controllers. Individual execution of simple reactive local policies to control local traffic, leads to the emergence of an overall traffic control of an entire network, with certain complex behaviours not defined a priori and unaware to the single controller. The primary difference between this approach and our architecture is that, while this approach relies on a reactive policy, we introduce deliberation into our controller, such that it will not only react to existing problems but can deliberate before execution when encountered with unforeseen problems. Our approach makes it possible for our architecture to work both centrally over an entire region or distributed on local traffic controls.

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.

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.

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.

The main difference between traditional and our self-management architecture is depicted in Figure 6.1. Traditionally, a control loop consists of three steps: sense, interpret and act [65]. 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 that are feedback into the system in other to keep the environment in the desirable state. Introducing deliberation in the control loop allows the system to reason and generate effective plans to achieve desirable goals. Enabling deliberative reasoning in UTC systems is important because of its ability to handle unforeseen situations that have not been previously learnt nor hard-coded into a UTC. This would help to reduce traffic congestion and carbon emissions.

In a quest to embed SM attributes into an urban traffic control system, we explore the feasibility of integrating an automated planning component into an urban traffic control system and using MPC to resolve some of the challenges envisaged in this architecture. The next section gives an overview of the architectural design of our approach.

6.2 The Self-Managing System Architecture

The key to the contents of the architecture in Figure 6.2 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 [20] and AI planning paradigm.

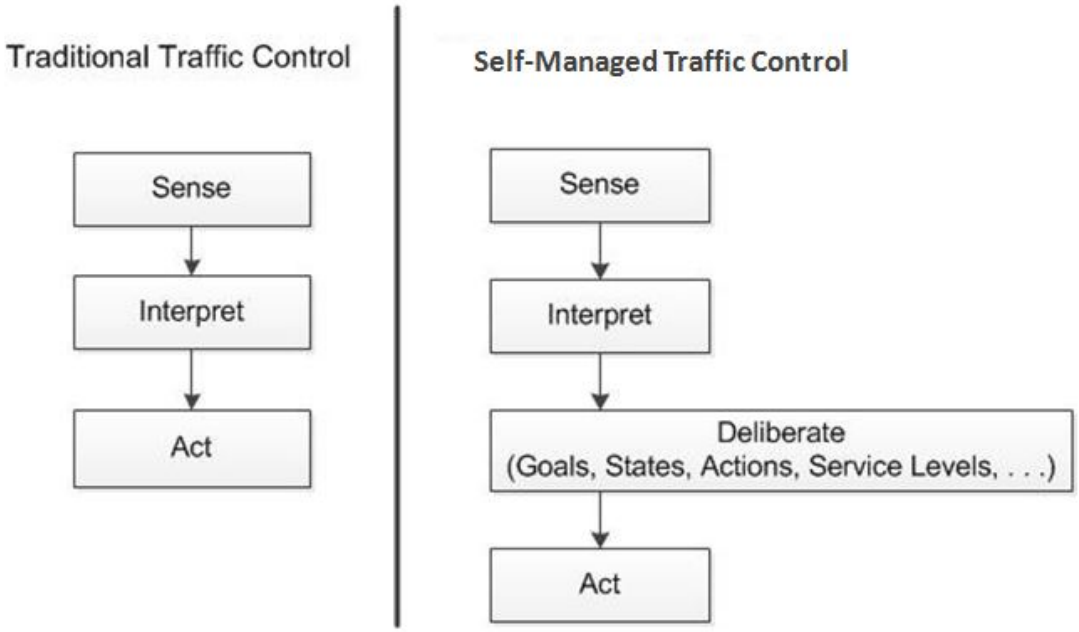

Figure 6.1: Comparison of traditional and deliberative controls in Urban Transport Systems.

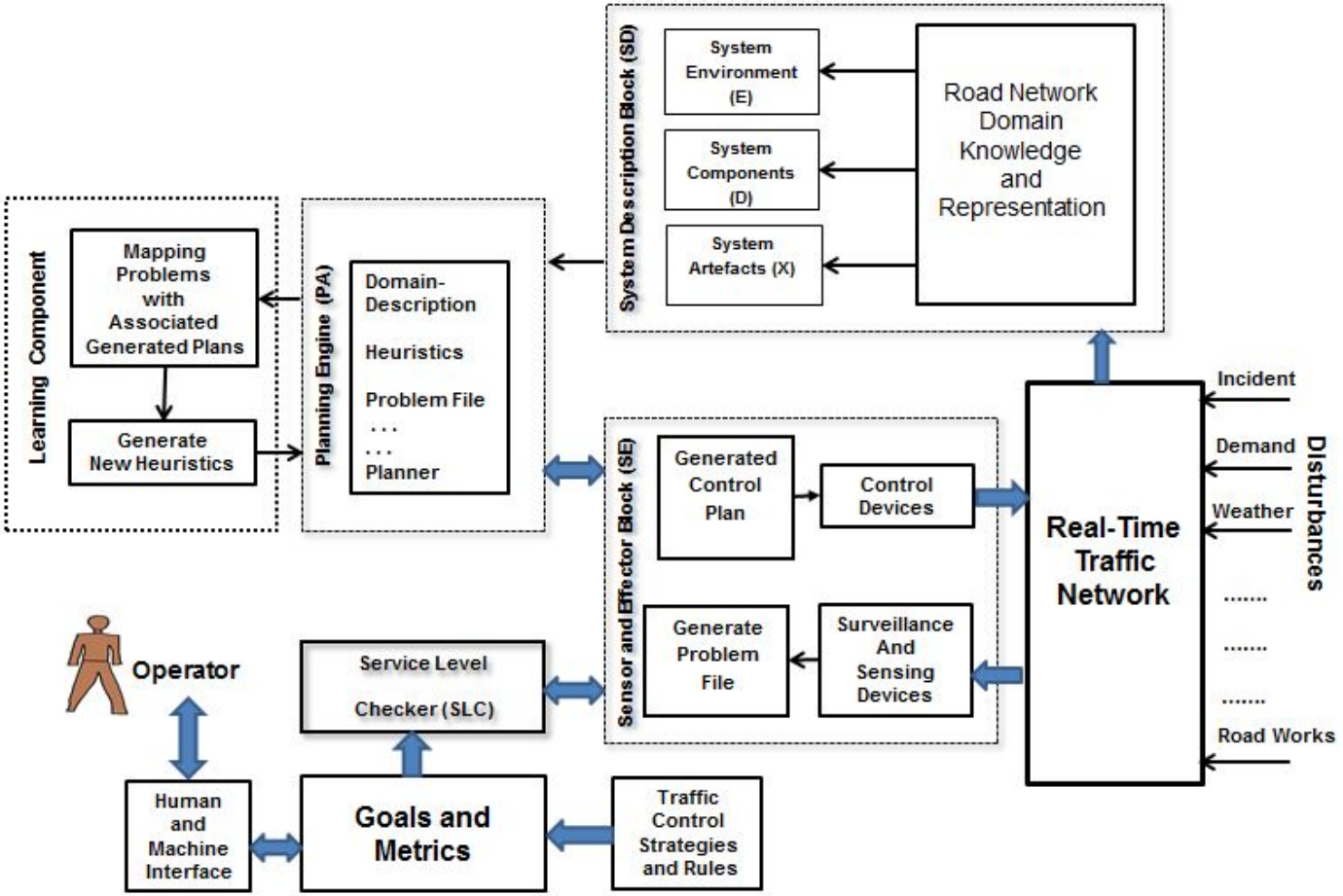

Figure 6.2: Diagram of a Self-Managing System Architecture

6.2.1 Road Traffic Network and Disturbances

Disturbances are factors that affect the proper functioning of a road network(detailed explanation in Chapter 3.0). Our perception of the traffic situation within a network of roads is that a network of traffic(detailed explanation in Chapter 3.0) will continue to operate as planned until a disturbance makes it act otherwise. Changes to the state of a traffic network are under the influence of various disturbances. For example, a sudden increase in the volume of traffic on a road network because of the end of a football game in a nearby arena becomes a disturbance to such road network, if the network has not been previously programmed or adapted to such a traffic change. In this work, a network of roads within Huddersfield town center area of United Kingdom was considered for our experimental analysis as detailed in the subsequent chapters.

6.2.2 System Description Block (SD)

A system needs both to be situationally aware, *and* have a knowledge model of its components, its environment and the relationship between them before a self-aware property can be created in such a system. 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 [100]. 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 elements, 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 a directed graph. Below is a snippet of the the connections between individual roads in our experimental example:

```
. . .
(:objects
    markStr
    //there exist a road in the network called mark street
....
  (: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. The capacity limit is the maximum number of the vehicles a road can contain at any giving period.

```
....
   (= (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.

6.2.3 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 devices 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 becomes the start of an information processing pipeline that 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 thesis [88]. 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 control systems to change their performance to a desirable state. Control devices play a 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 a 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 the desired 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 that provides a new plan.

6.2.4 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[108]
- 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 that aims to meet the system goals from the current state. The default planning technology to carry out this technique is AI Planning [108]. The plan produced is passed to the SE block for execution. However, it is well known from planning and execution systems [59] 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 that may consist of showing information about such alternative routes on variable message signs and optimising signal heads towards such route.

6.2.5 MPC Solution to Challenges in PA Block

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 to generate 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. 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 before the generation and execution of a control plan.

These two planning problems can be minimised by utilising a Model Predictive Control(MPC) approach [20] to supplement an automated planner.

MPC techniques help to reduce the situation whereby the environment has changed before 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 an entirely changed environment prior to plan execution.

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

This is achieved by breaking search exploration into 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 it takes 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 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 the fixed time or node count, the planner communicates the partial or completely generated control plan to the control network 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 produce a continuous curve like that of an MPC as shown in Figure 6.3. 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 compensates 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

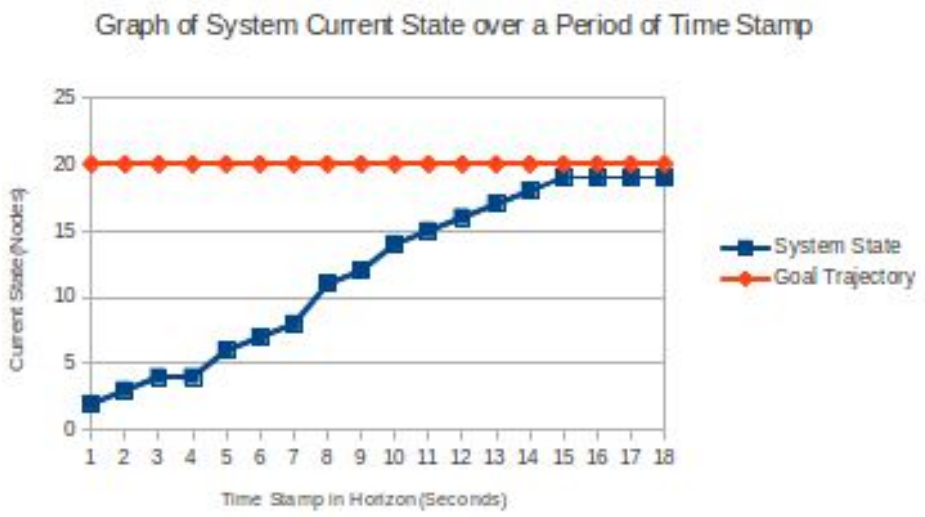

Figure 6.3: Illustration of a well guided real-time process curve.

(but not necessarily optimal) partial plan in the time allotted for plan generation.

6.2.6 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, utilising the idea of assigning blame or positive feedback over repeated trials [10; 46]. 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 [60]. 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 a particular period of time. One distinct 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 plans achieving a particular 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 in a state such that, if those features are seen again, the same plan can be re-used to achieve the same goals.

6.2.7 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'.

Formally, we can define an error function $\varepsilon : S \rightarrow \mathbb{R}_0^+$ which determines how far the current state of the system is from being 'ideal'. $\varepsilon(s) = 0$ 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 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 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 the 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.

6.3 Challenges

This architecture is presently implementable in systems that have a time delay. It cannot be implemented with real-time processes that require continuous changes in nanoseconds/microseconds. This is because 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 that may not be desirable. This could be as a result of a grossly inaccurate model that 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(state selection) for control plan generation should also encompass all significant dynamics and disturbances. Otherwise, performance may be reduced, and significant events may be unobserved.

The most important lesson learnt 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 that are tuned for UTC use. This will allow us to take advantage of the peculiarities of UTC application area, with the hope to create operationally successful planners.

6.4 Conclusion

This chapter describes our vision of self-management at the architectural level within a UTC management system. We created a self-management architecture made of several

blocks that 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 and MPC in enabling a self-management property in the systems architecture.

6.5 Summary

This chapter describes the design of a generic architecture which is aimed at using both AI planning and MPC approach in creating self-management properties in UTC. Our architecture consists of several blocks. This chapter describes each of the blocks in the context of an Urban Traffic Control Domain.

The next chapter demonstrates our efforts to create a hybrid planner that could work with this architecture to generate plans in the UTC domain. The hybrid planner is an integration of MPC approach into existing planning techniques to create a continuous planner that could reason with continuous nature of UTC domains in the present of multivariable environmental constraints.

Chapter 7

Design and Implementation

In the previous chapter, we described the design of a generic architecture which is aimed at using both AI planning and MPC approach in creating self-management properties in UTC. This chapter demonstrates our present efforts towards creating a hybrid algorithm for Urban Traffic Control PLANner(UTCPLAN), that could work within the self-management architecture to generate plans for certain continuous problems using UTC problem as an example.

7.1 Introduction

We propose the use of a Model Predictive Control(MPC) architecture in continuous planning to solve problems in domains that are modelled using variables whose values are changing continuously and to model processes and events that are internal and exogenous to our domain of concern. We design a hybrid algorithm that integrates MPC approach to existing planning techniques to create a continuous planner that could reason with the continuous aspect of a problem domain in the presence of multivariable environmental constraints.

AI planning(detailed in Chapter 2.) uses algorithms with heuristics to search for a feasible plan to solve a problem. These existing approaches have been an excellent

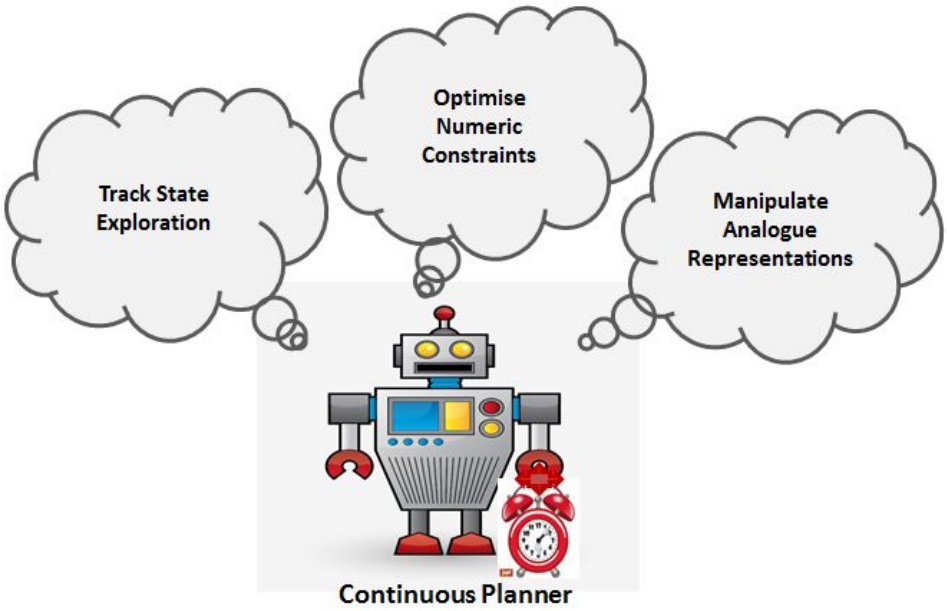

Figure 7.1: The Agony of a Continuous Planner

approach to generating satisficing plans, most especially in classical domains, but this method sometimes fails when problems become more complex [54]. MPC(detailed in Chapter 5.) on the other hand, has been known for its ability to optimise continuous processes in the presence of multi-variable constraints.

The continuous nature of MPC is combined with certain discrete numeric features of AI planning to create a novel algorithm called UTCPLAN. The algorithm is embedded into a planner that could reason with the set of parameters in a continuous domain and generates plans in a reasonable time. The planner design is broken down into two main categories.

1. The design of a declarative language for representing domain model and problem information.

2. The design of the planning algorithm that could reason with the set of parameters in the domain and problem.

Figure 7.2: The Planner as a Black Box

7.2 Planner Language Representation

UTCPLAN addresses planning domain descriptions that involve durative actions and metric resources with an extension of durative processes in the presence of events. The planner supports the modelling of continuous change brought about by exogenous processes and events. It can partly represent both the discrete and continuous numeric changes that occur as a result of its processes with influences on events.

The domain description language syntax and semantic of the planner is similar to *PDDL*+. *PDDL*+ proposed a realtime model that includes actions, exogenous events, and autonomous processes. Refer to the work of M. Fox and D. Long [58] for detailed explanations of the syntaxes and semantics of *PDDL*+ including the semantics on which implementations of state representation and state progression must be constructed.

Numeric fluents is introduced, to create reasoning about numeric values in domains. Though, the use of numeric fluents is restricted to basic numeric operations. Fluent are assigned instantaneous values at the initial states or by an action. This could be a definite value or an increase or decrease by value; or the use of basic arithmetic operators $(+,-,*,\div)$ to modify fluent base on arithmetic expressions. The durative action imposes an upper and lower bound constraint on the action operator. Continuous change in numeric value, according to some specified numeric functions, over the entire period of execution of an action are linearised. A differential equation of the form $\frac{dp}{dt} = f(P) + c$, specifies the effects on a variable p where P is the vector of state numeric variable, $f(P)$

is a mathematical function over these numeric variables, and c is a constant.

A set of linear continuous numeric effects is added to the component of each action a. $(increase(v)\ (*\ \#\ t\ k))$ which denotes an increases in v at the rate of k per unit of time.

$eff_{\vdash}n$ and $eff_{\dashv}n$ may additionally include durative parameter $?duration$ for both *start* or *end* effects of actions. This denotes the duration of the action, and could be written as:

$$\langle v, op, w \cdot v + k \cdot (?duration) + c \rangle, \text{ such that } op \in \{+=, =, -=\}, c, k \in \kappa$$

where w is a vector of constants and $?duration$ may have a constrain value that lie within a range of values, e.g. $(?duration \geqslant v1) \bullet (?duration \leqslant v2)$, for some numeric variables *v1* and *v2*.

The extension to the action component allows the value of a variable v to depend on the period elapsed from the beginning of the continuous effect acting upon it. The second extension to the durative effect implies that $?duration$ do not always need to be fixed, since the value of variables can now depend on the duration assigned to the action.

7.3 Planner Algorithm

The planner implements UTCPLAN algorithm to find feasible solution plan from problems description in a continuous domain. UTCPLAN algorithm uses an A^* search algorithm technique for node exploration from the initial state to the goal state. Node implies a point in a search space at which search frontiers or pathways intersect or branch.*State* information and transition are also stored in a node. Preconditions of the operators are checked against the proposition and numeric fluent at each node, if it is satisfied, the operator effect is applied and the new state becomes the current state. The resource optimisation problem within the domain model is solved at specific nodes dur-

ing node exploration. The search proceeds by applying each applicable operator to the current states in a receding horizon until a goal state is found or the node set is empty. The following subsection gives a description of UTCPLAN algorithm.

7.3.1 UTCPLAN Algorithm Preliminaries

This section contains definitions that are fundamental to the design of the planner algorithm.

Definition 1 (State) *A state S is a pair $\langle P,N\rangle$, where P is the set of atomic propositions and N is an assignment for numeric variable to values. A state describes what is true of some world at a snapshot of time assuming a Close World Assumption(CWA) on S.*

Definition 2 (Initial State) *Initial State $I = \langle P,N\rangle$, where P is the set of atomic propositions and N is an assignment of values to numeric variables that are true at the start of some planning problem.*

Definition 3 (Goal Condition) *A Goal Condition $G = \langle P,N\rangle$, where P is a set of atomic propositions N is a set of numeric variables. For a goal to be achieved in some state S values v satisfies some numeric constraints $v_L < v < v_U$ specified by G. Thus, S satisfies the goal condition if $p \in P$ for every proposition in S and $\exists v = c \in N : V_L < c < V_U$ for all v in N. Where c is a constant whose values is between the upper bound and lower bound of v.*

Definition 4 (Domain Model) *A Domain Model DM, consist of:*

- *A set of Propositions $\{p_1,...,p_k\} \in P$*
- *A set of numeric Functions $\{n_1,...,n_k\} \in N$*
- *A set of Resources $\{r_1,...,r_k\} \in R$ and*
- *A set of Actions $\{a_1,...,a_k\} \in A$*

```
(:action  switch_to_green
:parameters [at_junction, this_phase, from_road1, to_road2]
:duration dur
:cost
:precondition [(intersect at_junction this_phase from_road1 to_road2)]
                [(>(queueLenght (from_road1 0.0))
                (<(interuptLevel (to_road2 7.0))]
:effect ([(JflowActive at_junction this_phase from_road1 to_road2)])
        )
```

Figure 7.3: Sample of an Action Declaration

- *A set of Processes* $\{c_1, ..., c_k\} \in C$
- *A set of Events* $\{e_1, ..., e_k\} \in E$

Definition 5 (Action) *An instantaneous action is characterised by preconditions that must hold before the action and effects that hold after the action. The logical basis for actions is modelled as a collection of propositions P, and a vector of numeric variables v. Both P and v are manipulated and referred to by actions. The executability of an action is determined by their conditions. For example, the action switch to green has the precondition that the light is red and has the effect that the light is green. A durative action A has three sets of preconditions: The condition that must hold at start* $pre_{\Leftarrow}A$*, at the end* $pre_{\Rightarrow}A$ *and throughout the execution of the action* $pre_{\Leftrightarrow}A$*. Effect could be durative or instantaneous, instantaneous effects are bound to the start* $eff^{+}_{\Leftarrow}$ *and* $eff^{-}_{\Leftarrow}$ *or end of the action* $eff^{+}_{\Rightarrow}$ *and* $eff^{-}_{\Rightarrow}$ *where positive and negative denote the propositions added and deleted at the start and end of A respectively. Also numeric effect* $eff^{n}_{\Leftarrow}$ *and* $eff^{n}_{\Rightarrow}$ *are updated at the start and end respectively. An example of action declaration is shown in Fig 7.3.*

Definition 6 (Processes) *A process p comprises of a precondition, C, and a set of continuous effects, E, such that, if* $S \models C$ *then the continuous effects are active at state S.*

```
 (:process  Jtraffic_flow
:parameters [at_junction, this_phase, from_road1, to_road2]
:duration dur
:cost
:precondition [(JflowActive at_junction
                this_phase from_road1 to_road2)]
:effect ([(decrease(queueLenght (from_road1 (* #t flowrate)))
        (increase(queueLenght (to_road2 (* #t flowrate)))])
            )
```

Figure 7.4: Sample of an Process Declaration

For instance, the inflow process of vehicles V to a road R through a junction J. This process has a precondition that a given phase at junction J is active that is 'Green' and that the road_use_level of R less than the road-capacity-level; and the constraint that J is a connected inflow junction to road R. Once R is filled or blocked, an event is triggered that stops the process. Inflow process has the effect of increasing R traffic level at the rate of flow of V as shown in figure 7.4. At any point in time, the derivative of traffic level in R is the sum of the rates of the active inflow process processes.

Definition 7 (Event) *An event e is a state transition (C, E), where C is a declaration expressing the triggering condition of e and E describes the effects of the event. The event e is triggered in any state S such that* $S \models C$. *A new state* s' *is results upon the application of the effect E such that* $s' \dashv E$.

An ordered finite set of event, $EC_S = \{e_1, ..., e_n\}$, *with conditions* $\{c_1, ..., c_n\}$, *will cascade triggered in a state S , such that for every event* e_i *in* EC_S, *either* $S \models c_i$ *or the subset of events ordered before* e_i *achieves a state* s' *such that* $s' \models c_i$. *For example an event 'upstreamFilled' to be triggered, it requires the estimated number of vehicles on such road to be equal or greater that the road capacity limit of such road as shown in Figure 7.5.*

Definition 8 (Operators) *Given a set of proposition* $P_{(s)}$ *and numeric fluents* $N_{(s)}$ *as describe above, a numeric operator* δ *is a pair* $\langle pre(\delta), eff(\delta) \rangle$ *given that:*

```
 (:event  upstreamFilled
:parameters [at_junction, from_road1, to_road2]
:duration dur
:cost
:precondition [(>=(queueLenght (to_road2 capacity_of_road2)))]
:effect ([(assign(queueLenght (to_road2 capacity_of_road2))
            (assign(interuptLevel (to_road2 7.0))])
        )
```

Figure 7.5: Sample of an Event Declaration

- *The condition for applicability* $pre(\delta)$ *of an operator* δ *consist of:*
  - *a proposition or set of propositions* $pre_{prop}\delta$ *define over P*
  - *a numeric or set of numeric comparisons* $pre_{num}\delta$ *of the form* $(exp\{>,\geq,<,\leq,=\}exp')$.
- *The effect of an operator* $eff(\delta)$ *consists of:*
  - *an additional proposition* $eff^{+}(\delta)$ *produced and a deleted proposition* $eff^{-}(\delta)$ *removed after the operator execution.*
  - *set of numeric operations* $eff^{+}_{num}(\delta)$ *in the form* (n,op,exp)*, where* $n \in N$ *is the numeric fluent affected by the operation op, which could be one of* $\{+=,=,-=\}$.

In this definition, the arithmetic expression exp and exp' *involves variables from N. These are recursively defined constants, numeric fluent or an arithmetical combination of* $\{+,*,-,/\}$ *among expressions.*

Definition 9 (Operators Applicability) *An operator* δ *is said to be applicable in a state s iff its propositional and numeric preconditions are satisfied by s. That is,*

- $pre_{prop}(\delta) \subseteq P_{(s)}$ *and*
- $pre_{num}(\delta)$ *must be valid(i.e equal or in range of values) in all n where* $n \in N_{(s)}$

Definition 10 (Plan) *A plan comprises of a sequence of actions, with associated processes, that transforms the initial state into a state satisfying the goal state, respecting all the constraints in the conditions imposed.*

Given a continuous planning problem $\Psi = \{I, G, DM\}$ *where, I is the initial state, G is a set of goal conditions and DM is a set of operators. A solution for* Ψ *is a total ordered set of operators from* δ*, such that the ordered sequence of execution of these operators transforms I into a state where G is satisfied*

Algorithm 1 Top level algorithm of UTCPLAN

Input: I,G,DM.
Output: Plan.

```
1:  n := (I,R)
2:  Q := {n}
3:  repeat
4:      n.ℜ := OptimiseNumerics(n, DM, ℘)
5:      i := 0
6:      while Q ≠ {} and i < N_c do
7:          n := PopQueue(Q)
8:          N := Expand(n)
9:          if SolutionFound(N) ≠ TRUE then
10:             PushQueue(N,Q)
11:             i := i + 1
12:         end if
13:     end while
14:     n.℘ := UpdateDynamicInformation(n, N_p, N_c, DM)
15: until Q = {} or SolutionFound(Q)
```

7.3.2 Top level algorithm of UTCPLAN

The input to the planner is the initial state, goal condition and the domain model as defined in the preliminary definitions. The planner output is a sequence of operators (actions and processes) as described in the preliminary definitions. Line 1 initialises the initial state with the associated resource declarations at the initial state. The initial state is set to the initial node in Line 2. and stored at the first node of a priority queue.

Line 3. to 15. repeats the search and optimisation processes from the current node until *solutionFound* flag is true or the node is empty.

Line 4 uses the state at the node with it associated resource values and constraint to optimise the numeric fluents in the domain model(DM) assumed the dynamic information $\wp$ at the initial state is zero. Line 5. initialises the control horizon N_c counter to zero value.

While the priority queue is not empty and the horizon control value N_c is not exceeded, a new node is *popped out* from the queue and expanded in Line 7. to 13. The search terminates when *solutionFound* flag becomes true or the node is empty. Otherwise, the expanded node is **pushed back** to the priority queue in Line 10. and the horizon value is incremented in Line 11.

The current node is expanded in Line 8. with the application of an operator, details of the expansion algorithm is given in Algorithm 2. Line 9. checks the expanded state to see if the goal condition has been met. Definition of the goal condition has also been defined in the preliminaries given in the previous section. The search terminates when *solutionFound* flag becomes true or the node is empty. Otherwise, the expanded node is **pushed back** to the priority queue in Line. 10 and the horizon control value is incremented in Line 11.

Whenever the fix control horizon N_c is exceeded, the UpdateDynamicInformation (UDI) function is called from the current node in line 14. The EDI function extracts stored numeric changes and information from the planner, generates dynamic prediction values $\wp$ over the period of horizon N_p. $\wp$ is sent to OptimiseNumeric function to optimise the constraints in the domain DM at node n . The new controlled value is updated in the node. The horizon window N_p is reinitialise for another fixed period and the search continues. The search and optimisation processes are repeated in Line 15. until *solutionFound* flag becomes true or the node becomes empty. The next subsections explain the procedures in the top level algorithm.

7.3.3 Expand Search Node(n)

The current node n is expanded by selecting the appropriate operator that satisfies the condition at a node. The effect of the operator changes the state at a node from n into another state N as show in Algorithm 2. The procedure for the application of an operator, grounded process or an event is explained in Algorithm 3, 4 and 5 respectively.

Algorithm 2 Procedure for Expand(n)

Input: n
Output: N

$N := \{\}$
for all operators $o \in DM$ **do**
  $O := \{o' | o'$ is an instantiation of o, and $n.s$ makes $o'.pre$ true $\}$
  for all $o' \in O$ **do**
    $s1 :=$ apply o' to $n.s$
    $E := \{e' | e'$ is an instantiation of some $e \in DM$, and $s1$ makes $e'.pre$ true $\}$;
    $s2 :=$ apply all events in E sequentially to $s1$
    $P := \{p' | p'$ is an instantiation of some $p \in DM$, and $s2$ makes $p'.pre$ true $\}$
    $s3 := s2$ with all processes P started
    $N := N \cup \{(s3, n.r)\}$
  end for
end for

7.3.4 Action Application

Definition 11 (Apply Action) *Given a state s and an action a, such that a is applicable in s, the application of a in s, denoted by $s[a]$ to produce a new state s' is as shown in Algorithm 3.*

An action is an instance of a ground operator within the domain whose execution takes the state to a state that is closer to the goal condition. It preconditions could be logical, or both logical and numeric inequalities and its effect are usually logical and sometimes numeric update to the current state where the action is executed.

For instance, the action 'switch to green' in Fig 7.3 has a logical precondition that the two roads must be intersected at the junction and must be sharing the same green

phase. It also has numeric preconditions such as the interrupt level of the connected road must be less that interrupt level seven. The effect of this action changes the logical state of the phase at the junction of the two roads to be active, which subsequently starts a process at that junction in the next node. The planner selects the best sequence of action that could take the current state at any particular node to a state that satisfies the goal condition.

Algorithm 3 Action Application

Input: s, a
Output: s'.

1: s' is initialised to be s;
2: All propositions present in $eff^{+}a$ that are not already in s are added to $P_{(s)}$
3: All proposition present in $eff^{-}a$ are removed from $P_{(s)}$
4: All numeric fluent f such that $(f, op, exp) \in eff_{num}(\delta)$ is updated and modified according to the defined op and exp involved.
5: All state $s \in S$ obtained by a non applicable operator is undefined and does not satisfy any condition.

7.3.5 Simulate Process

Definition 12 (Simulate Process) *Given a state s and a ground process c, such that c is applicable in s, the application of c in s, denoted by* $s[c^{+}]$ *to simulate continuous numeric changes in s for a period of time is as shown in Algorithm 4.*

A grounded process within the domain runs for a period of time once it is initiated within the search node. The time is discretised into single step counts(E.g. $t = 1, 2, 3...t_n$) where t_n is the duration of simulation of the process. Processes are started as a result of an action initiating the process or an event triggering the start of a process. The preconditions could be logical or numeric inequalities and their effects are also numeric update to the current state where the process is activated.

An example of a process initialisation is the resulting effect of an action "switch-to-green" in a domain. This action effect could lead to a process flow of queues from one

road to another at the rate of flow of traffic in that junction for the duration of active green phase at the junction as shown in Fig 7.4. The process would continue to run until the specified simulation time elapses or there is an internally generated event that halt the process. Once the duration of computing the process elapse, the current state of the process is also pushed to the priority queue.

Algorithm 4 Simulate Process

Input: s, c
Output: s'.

1: initialise process duration time $count = dur$
2: **repeat**
3: All numeric fluent f such that $(f, op, exp) \in eff_{num}(c)$ is updated and modified according to the defined op and exp involved
4: Time #t and other primitive numeric variables are updated
5: **until** event e is triggered or dur exceeded.

7.3.6 Event Application

Definition 13 (Apply Event) *Given a state s and an a ground event e, such that e is applicable in s, the application of e in s, denoted by* $s[e]$ *produces a new state* s' *as shown in Algorithm 5.*

The application of an event is similar to an *action* operator, except that, whereas an action **may** occur if its preconditions hold, an event, on the other hand, **must** occur if its precondition hold. An *event* in the domain could be internally triggered from within a *process*, or outside the control of a *process*. Internally triggered *event* are *interrupts* that are activated while a *process* is running, it preconditions are usually numeric inequalities and their effect are also numeric assignments. These numeric assignments are set as *preconditions* for some *actions* in the domain. This means that the *interrupts* tell the planner to execute an emphaction that could change the emphstate of the system or flag a display.

An example of *event* is to manage the constraint of traffic spill-over at junctions during rush hour as shown in Fig 7.5. It has a *precondition* to check the capacity of the connected road during the process of traffic flow at a junction. The effect of this event stops the current running process from transferring queue to the upstream road. This is achieved by an interrupt trigger that halts the process and pushes the current state of the node to the priority queue node.

Externally triggered *event* are a result of interaction between domain objects. The preconditions could be logical or numeric inequalities and their effects are also numeric update to the current state where the event is activated. An example of an external event is the activation of connectors that link two separate roads. Once the condition for the connector is satisfied, the queue from the previous road flows to the connected routes. This is outside the control of a junction, but the ripple effect of such event (traffic flow) affects the queues at downstream of the junctions. The different between this connecting event and an action is that once the event precondition is satisfied, it has to be activated, computed and updated to the current state, however, an action might only be selected if it necessary get the state closer to the goal state.

Algorithm 5 Apply Event

Input: s, e
Output: s'.

1: s' is initialised to be s;
2: All proposition present in $eff^{-}e$ are removed from $P_{(s)}$
3: All propositions present in $eff^{+}e$ that are not already in s are added to $P_{(s)}$
4: All numeric fluent f such that $(f, op, exp) \in eff_{num}(\delta)$ is updated and modified according to the defined op and exp involved.
5: Time #t and other primitive numeric variables are updated

7.3.7 Optimise Numeric Fluents(n, DM, $\wp$)

The input to the *OptimiseNumerics* is the current node n, the predicted dynamic traffic information $\wp$ and information from the domain model DM. This resource optimisation

sub procedure works as a planning satisfiability(SAT) problem (originally used in [4; 126]). The conflicting resource parameters with their corresponding constraints values are translated at a given search node into a linear programming problem. The problem is them optimised by using an embedded constraint solver. The optimised values are updated at the point of call to this sub procedure. For example, this sub procedure will generate a new set of controlled variable($g(k)$) at a search node, for the state vectors ($x(k)$) during search space whenever the value of the control horizon N_c is exceeded. It optimises all the updated dynamic traffic information $\wp$ and domain constraints in DM with the objective of minimising the state variables ($x(k)$) on the controlled variable ($g(k)$).

7.3.8 Update Dynamic Information(n, N_c, N_p, DM)

The input to the *UpdateDynamicInformation* is the current node n, the horizon prediction value N_p, the value of control horizon window N_c and information from the domain model DM. This subprocedure stores the past trends of numeric changes within the nodes as search progresses within search space.

Whenever the horizon control value N_c is reached, past numeric fluents are retrieved from the planner. This numeric information is used to generate a dynamic prediction table over the period of prediction horizon count N_p. The generated values are parse to an optimiser to compute the best control values for the next set of alteration taking into considerations all the constraints in the domain. The new values are updated dynamically while the search space progresses for another period of control horizon N_c. This process is repeated once a call is made to the procedure until the goal conditions are satisfied.

For instance, assuming N_c is set to 500 node count and N_p is set to two minutes. The planner would keep tract of the node counts and retrieve past numeric fluent at every 500 node count. It will use these numeric information to generate a dynamic prediction

table of changes in numeric values over the period of prediction horizon of two minutes. The generated values will be parse to an optimiser to compute the best green split value for subsequent search period taking into considerations all the constraints in the domain. The new green split would be altered and updated dynamically at the node that initiates the call while the search space progresses for another period of 500 node count (control horizon N_c).

Algorithm 6 UpdateDynamicInformation (n, N_p, N_c, DM)

Input: n, N_p, N_c, DM
Output: $n.\wp$

1: $Q := \{n\}$
2: i := 0
3: **repeat**
4: $n.\hbar =$ store numeric changes
5: $i := i + 1$ increment i
6: **if** i equals N_c **then**
7: $n.\Im =$ extract numeric changes from $n.\hbar$
8: n.$\rho =$ generate prediction values from $n.\Im$ over a period of N_p
9: $n.\Re =$ OptimiseNumerics(n.ρ, DM, $\wp$')
10: $n.\wp = n.\wp \bigcup \{(n.\rho, n.\Re)\}$
11: i := 0
12: **end if**
13: **until** $Q = \{\}$ or SolutionFound(Q)

7.4 Planner Implementation

The planner is implemented in Netbeans Java 8.0. The domain and problem representation (traffic description) are also developed with advanced Java to facilitate smooth data transfer between planner and network information description. The experiment was carried out on an Intel(R) Core(TM) i7-4702MQ CPU, 2.20GHz, 16GB RAM with a 1.5 Terabyte memory capacity.

The network traffic problem comprises of twelve connected roads, two linked roads and three junctions as shown in Appendix F. The domain model contains actions, pro-

cesses and events that apply to the domain as shown in Appendix E.

The input to the planner includes the state variables represented by the queueing in meters of road at any instance of time; The control variable represented by the green split duration of the traffic control; declarative description of the map of the sampled area; The available actions operators, ground processes and event in the domain and the problem situation of the domain represented by the initial and goal state of the traffic situation.

The goal is to reduce traffic congestions on roads and to divert traffic in the event of any unexpected situation on roads. Technically, the goal is to reduce queue length on roads from source to sink at the shortest period of time using the minimum number of actions and processes to achieve the goal condition. The traffic problem supplied to the planner consist of a declarative representation of the road network; The initial queues form the source and the goal conditions for the queues at the sink are also supplied at the initial state.

The node exploration in search space compensates for the MPC approach as described by algorithm 1. The search proceeds by applying each applicable operator to the current state and put each resulting state into a priority queue. The horizon value N_p is fixed at 10 seconds count. Different control values were used for sampling during node exploration. A control window N_c of three node count is selected, because of its suitability for this domain compared to other control windows. Though, sampling at every node count give a better control result, but sampling at every node counts requires more computational time compared to sampling at a multiple of three node counts. Both the N_c and N_p values are chosen after multiple tests to decide a suitable horizon value for the planner.

Optimising all constraints by calling the numeric solver at every node makes it difficult or impossible to generate a plan at the shortest possible time, because of the complexities of numeric constraint in a UTC domain. Thus, changes in large numeric values in a node, which might have occurred due to changes in numeric values from

process simulations, are detected and updated during every sampling time(N_c). Other changes in primitive numeric expression are computed and evaluated by the numeric capacity of the planner at every node.

7.4.1 Planner Implementation Assumptions

The network state maintains the numeric counts (continuous approximation) of queue length at different abstractions of locations based on the following: route (R) explored by the planner during search space; queue (Q) denoting the numeric value of each road object at any instance of time; Source (Sc) which represents the entering road to the networks and sink (Si) which represents the exit roads. Vehicles originate from the source, passes through roads, connectors and junctions, then end up in sink upon reaching the destination.

At each time instant, a road may be active or inactive. For the instants during which a road is active, vehicles are assumed to move on that road at the flow rate of the road (veh/s). We assumed the flow rate of the roads were known and fixed at the initial state. When a link is inactive, it means there is no vehicles movement through the road and flowrate of such road is zero.

Each of the junctions has two phase (1 and 2). Traffic can move from north to south or from east to west at junctions. Two conflicting roads cannot be activated at the same time at a junction. The domain model, incorporate declarative descriptions of grounded *event* that monitors the movement of traffic within linking roads. The planner select the appropriate green phase duration to controls the traffic of roads connected at a junction.

All dynamic inputs, such as turning rates, with an exception of the state variables and controlled variables, are assumed to be constant. The flow rate of individual junctions is also assumed to be constant. Rate of flow of vehicles is represented as a unit value per seconds of time (veh/sec).

We assume that time evolves in discrete steps $(k = 0, 1, 2, ..., k_n)$ corresponding to

traffic light cycles. We denote the network state vector by x(k) and the control vector is denoted by g(k). We assume we cannot control drivers behaviour; thus, we only control the green split (the controlled variable). We also assumed that the traffic flow dynamics are fully defined and included in the domain file.

We consider a linearised version of the quadratic problem that simplifies real-time calculations. The objective is to minimise the total numbers of vehicles in a queue on all roads at any instance of time within the traffic network. See Chapter 5. for details of domain constraints and objective function. The optimisation problem is solved using an embedded API to Java called *OjAlgo Solver*.

OjAlgo is a solver that contains LP (Linear Programming), QP (Quadratic Programming) and MIP (Mixed Integer Programming) solvers. ojAlgo is Open Source Java code that has to do with mathematics, linear algebra and optimisation. The core of ojAlgo is a linear algebra framework complete with various matrix decompositions using highly efficient multidimensional arrays. It can use double, BigDecimal or ComplexNumber as matrix elements. Though, there are lots of commercial solvers that are better than ojAlgo; however, ojAlgo is chosen due to it open source advantage. It also a simple framework to model problems using standardised expressions as well as the ability to read MPS (Mathematical Programming System) files.

Linearised methods often led to suboptimal solutions and could not consider the limits of some constraints exhaustively. Therefore, exploring more complex optimisation solution that can scale better in preferred for future purpose. The main objective of this implementation is not to scale the output metrics, but to investigate the feasibility of using our UTCPLAN approach in this domain of interest(UTC).

7.5 Conclusion

We propose an integration of MPC approach into existing planning techniques to create a hybrid algorithm(UTCPLAN) for a continuous planner. This algorithm works with a

blend of planning and constraint optimisation to optimise a given traffic flow problem from initial state to a state that satisfies the goal conditions. The node exploration in search space of UTCPLAN is modified to compensate for the MPC approach. This is achieved by introducing a feedback strategy into the search space to sample the state at a node whenever the value of controlled horizon window is exceeded. Constraints are optimised over a future horizon value while the optimisation effect is only used within the next control horizon window. This repetition is continued within the search space until the goal conditions are satisfied. The algorithm is implemented in Java Netbeans 8.0 and tested with a declarative description of an urban traffic network.

7.6 Summary

This chapter describes the design and implementation of a planner that can input expressive domain descriptions containing continuous processes, events and actions and produce solution plans through the integration of MPC with AI search - based planning techniques. The next chapter explains the result and evaluation of our implementation.

Chapter 8

Evaluation and Results

The main evaluation criterion is to show that UTCPLAN can accept inputs expressive domain descriptions containing continuous processes, events and actions and output solution plans through the integration of MPC with AI search - based planning techniques. This evaluation is measured by creating an expressive declarative description of a UTC domain with traffic flow problems to test if UTCPLAN can generate execution plans that can transform any given traffic condition to a state that meets the specified goal conditions.

The experimental traffic network(domain) is designed to have more than one connected junctions in other to test the centralise reasoning of UTCPLAN to manage upstream and downstream of traffic from connected road to the junction. This also allows us to test the feasibility of junction to junction traffic relationship within the network. Each junction is designed to have more than one signal phase to test the ability of UTCPLAN to split the green time between the two phases at a junction base on the current state of the queue length associated with each road at the junction. The network model also has connected roads linking other roads without a signalled junction, in other to test the ability of UTCPLAN to reason with the dynamic state of those connected roads that are not directly linked to a junction in the network. Figure 8.1 shows a sample output plan generated by UTCPLAN from an urban traffic control problem of Appendix F. and

a UTC domain declaration of Appendix E.

Having achieved the main evaluation criteria, we also evaluate the effectiveness of the embedded MPC approach in the UTCPLAN algorithm to optimise traffic flow during changes in the traffic situation. We test the performance of the planner base on its ability at controlling the signalled junctions to accommodate for the changes in the traffic situation.

To achieve this, we create two signalled situation in our experiment:

Fixed The signal at the junctions are fixed from initial state to goal state, the planner cannot alter the signal duration during search space. It would reason with the domain and generate solution plans using the fix signal value at the junction.

Controlled The signal is controlled by the planner. In this case, the signal is set at the initial state, but the planner can change the signal value during search space using the embedded MPC approach in UTCPLAN to optimise the green phase at junction base on traffic demands within the network.

We investigate the speed of UTCPLAN on selected volume of traffic to test the performance of UTCPLAN during traffic congestion. This is achieved by increasing the queue length on roads and comparing the time taken to generate plan when the signal is fixed or controlled. We also evaluate the quality of plans generated by comparing the number of actions and processes in the fix signalled plan to the controlled signalled plan.

8.1 Results

The output of the planner is a sequence of steps that takes the problem from it initial state to a state that satisfies the goal condition. Figure 8.1 shows a sample of plan generated by UTCPLAN to solve an urban traffic control problem of Appendix using a domain description of Appendix. The plan contains the sequence of action operators

needed to optimise traffic flow within an urban traffic network, until the goal condition is satisfied. The plan also shows the status of processes at junctions and between connected roads. The status of event are not displaced in the plan, but the effect of events on processes within the connected roads is reflected in the plan(example : [process] Rtraffic flow).

8.1.1 Evaluation Policy

The planner output contains the following information about the plan:

Total Time Total time to generate a new plan

Makespan Total plan execution time

Plan Length Total length of the output plan which includes the total number of processes, actions and some events effects in the plan. This also gives the quality of the output plan, the shorter the length the better the plan.

NodeCount Number of nodes explored in search space

No. of Actions Total number of actions required to achieve the goal condition

No. of Processes Total number of processes initialised within the plan

In the field of AI planning, the total time taken to generate a plan is a metric that shows the efficacy and speed of the planner. The total time depends majorly on the planner algorithm. It is also dependent on some other factors such as the language used to implement the planner and the hardware configuration of the system that the planner resides on. The faster it is to achieve the goal condition the lesser the total time to generate a plan and vice versa. The total time taken to generate a plan is an essential criterion for the evaluation of planners in AI planning. A planner is effective in a domain of problem if the total time to generate a plan for problems in that domain remains steady and stable. However, if the total time to generate a solution in a domain of

```
MAKESPAN        OPERATOR        STATUS                          PARAMETER                                       DURATION
0.0     switch_to_green                                         [JunctionA, phase1, nLNorth, nLSouth]           5.0
5.0     switch_to_green                                         [JunctionA, phase2, jWNorth, jWSouth]           5.0
10.0    switch_to_green                                         [JunctionB, phase2, brStr, lDSouth]             5.0
15.0                            [process] Jtraffic_flow         [JunctionA, phase1, nLNorth, nLSouth]           5.0
20.0                            [process] Jtraffic_flow         [JunctionA, phase2, jWNorth, jWSouth]           5.0
25.0                            [process] Jtraffic_flow         [JunctionB, phase2, brStr, lDSouth]             5.0
40.0    switch_to_green                                         [JunctionA, phase1, nLNorth, nLSouth]           5.0
45.0    switch_to_green                                         [JunctionA, phase2, jWNorth, jWSouth]           5.0
50.0    switch_to_green                                         [JunctionB, phase2, brStr, lDSouth]             5.0
55.0                            [process] Jtraffic_flow         [JunctionA, phase1, nLNorth, nLSouth]           5.0
60.0                            [process] Jtraffic_flow         [JunctionA, phase2, jWNorth, jWSouth]           5.0
65.0                            [process] Jtraffic_flow         [JunctionB, phase2, brStr, lDSouth]             5.0
70.0                            [process] Rtraffic_flow         [nLSouth, wDStr]                5.0
75.0                            [process] Rtraffic_flow         [lDSouth, pTStr]                5.0
90.0    switch_to_green                                         [JunctionA, phase1, nLNorth, nLSouth]           5.0
95.0    switch_to_green                                         [JunctionA, phase2, jWNorth, jWSouth]           5.0
100.0   switch_to_green                                         [JunctionB, phase1, wDStr, lDSouth]             5.0
105.0   switch_to_green                                         [JunctionB, phase2, brStr, lDSouth]             5.0
110.0   switch_to_green                                         [JunctionC, phase1, pTStr, bmStr]               5.0
115.0                           [process] Jtraffic_flow         [JunctionA, phase1, nLNorth, nLSouth]           5.0
120.0                           [process] Jtraffic_flow         [JunctionA, phase2, jWNorth, jWSouth]           5.0
125.0                           [process] Jtraffic_flow         [JunctionB, phase1, wDStr, lDSouth]             5.0
130.0                           [process] Jtraffic_flow         [JunctionB, phase2, brStr, lDSouth]             5.0
135.0                           [process] Jtraffic_flow         [JunctionC, phase1, pTStr, bmStr]               5.0
140.0                           [process] Rtraffic_flow         [nLSouth, wDStr]                5.0
145.0                           [process] Rtraffic_flow         [lDSouth, pTStr]                5.0
160.0   switch_to_green                                         [JunctionA, phase1, nLNorth, nLSouth]           5.0
165.0   switch_to_green                                         [JunctionA, phase2, jWNorth, jWSouth]           5.0
170.0   switch_to_green                                         [JunctionB, phase1, wDStr, lDSouth]             5.0
175.0   switch_to_green                                         [JunctionB, phase2, brStr, lDSouth]             5.0
180.0   switch_to_green                                         [JunctionC, phase1, pTStr, bmStr]               5.0
185.0                           [process] Jtraffic_flow         [JunctionA, phase1, nLNorth, nLSouth]           5.0
190.0                           [process] Jtraffic_flow         [JunctionA, phase2, jWNorth, jWSouth]           5.0
195.0                           [process] Jtraffic_flow         [JunctionB, phase1, wDStr, lDSouth]             5.0
200.0                           [process] Jtraffic_flow         [JunctionB, phase2, brStr, lDSouth]             5.0
205.0                           [process] Jtraffic_flow         [JunctionC, phase1, pTStr, bmStr]               5.0
210.0                           [process] Rtraffic_flow         [nLSouth, wDStr]                5.0
215.0                           [process] Rtraffic_flow         [lDSouth, pTStr]                5.0
230.0   switch_to_green                                         [JunctionA, phase1, nLNorth, nLSouth]           10.0
240.0   switch_to_green                                         [JunctionA, phase1, nLNorth, nLSouth]           10.0
250.0   switch_to_green                                         [JunctionA, phase2, jWNorth, jWSouth]           10.0
260.0   switch_to_green                                         [JunctionA, phase2, jWNorth, jWSouth]           10.0
270.0   switch_to_green                                         [JunctionB, phase1, wDStr, lDSouth]             10.0
280.0   switch_to_green                                         [JunctionB, phase1, wDStr, lDSouth]             10.0
290.0   switch_to_green                                         [JunctionB, phase2, brStr, lDSouth]             10.0
300.0   switch_to_green                                         [JunctionB, phase2, brStr, lDSouth]             10.0
310.0   switch_to_green                                         [JunctionC, phase1, pTStr, bmStr]               10.0
320.0   switch_to_green                                         [JunctionC, phase1, pTStr, bmStr]               10.0
330.0                           [process] Jtraffic_flow         [JunctionA, phase1, nLNorth, nLSouth]           10.0
340.0                           [process] Jtraffic_flow         [JunctionA, phase1, nLNorth, nLSouth]           10.0
350.0                           [process] Jtraffic_flow         [JunctionA, phase2, jWNorth, jWSouth]           10.0
360.0                           [process] Jtraffic_flow         [JunctionA, phase2, jWNorth, jWSouth]           10.0
370.0                           [process] Jtraffic_flow         [JunctionB, phase1, wDStr, lDSouth]             10.0
380.0                           [process] Jtraffic_flow         [JunctionB, phase1, wDStr, lDSouth]             10.0
390.0                           [process] Jtraffic_flow         [JunctionB, phase2, brStr, lDSouth]             10.0
400.0                           [process] Jtraffic_flow         [JunctionB, phase2, brStr, lDSouth]             10.0
410.0                           [process] Jtraffic_flow         [JunctionC, phase1, pTStr, bmStr]               10.0
420.0                           [process] Jtraffic_flow         [JunctionC, phase1, pTStr, bmStr]               10.0

Total Makespan:  430.0
Number of Actions Operators in PLAN:  26
Number of executed PROCESSES in PLAN:  32
Total Output Steps:  66
Node Count:  10

Timings in milliseconds:

Preprocessing Time is   77
Total Searching Time is   1382
Total Planning Time is   1459
Total Planning Time in Seconds:   ~ 1seconds
```

Figure 8.1: A sample plan generated by UTCPLAN

problem is astronomically increasing with an increase in the complexity of the problem, it means the planner might get stuck during certain problem situation in such domain.

The total length of a plan is the number of the steps it will take to get to a goal condition from an initial problem situation. The total length of a plan for a given problem varies from planner to planner. The shorter the length of the generated plan, the better the quality of the plan. The number of Actions and processes in the plan, contribute to the total length of the plan. The lesser the number of actions and processes needed to achieve a goal state the better the quality of the plan for such problem domain.

To investigate the applicability and effectiveness of UTCPLAN, we use three evaluation criteria for comparison: total time taken to generate a plan; the average number of processes used and the average number of actions initiated in the plan. We did not consider the makespan in this criteria because this implementation does not include a scheduler for makespan optimisation in the plan. Thus, using makespan as a major metric would not be suitable as criteria for evaluation of the planner.

We consider the performance of the planner with **fixed** and **controlled** signal value. Fixed duration means the duration of the green split is fixed at the initial state of the problem and would be the same throughout the planning time. The planner accepts the initial fixed timing from the problem file and uses this during the entire search space to generate the best combination of steps needed to achieve a goal condition that satisfies the problem situation. Controlled duration means that the duration is fixed at the initial state, but subject to changes whenever the planner anticipates a better optimum green phase than the fixed value during search space. In this case, the planner stores the past trends of numeric changes within the roads as search progress within search space. The past information of individual road input along with some dynamic information is retrieved from the planner. This numeric information is used to generate a dynamic prediction table over a fixed period of time. The generated values are sent to an optimiser to compute the best green split values for the next set of alteration taking into considerations all the constraints in the domain. The duration of green split at a

junction is updated dynamically during search space while the search progresses. This approach helps the planner to track and maintain changes in numeric fluents during search space.

We generated different traffic situation by increasing the percentage of road queues to create heavier traffic flow in the experiment. The simulation results for fixed time duration and controlled strategy are presented in Table 8.1. Given that x_2 is the new average value and x_1 is the previous average value, the percentage change in value $y\%$ is measured by equation 8.1 and recorded in table8.1.

$$y(\%) = \frac{x_2 - x_1}{x_2} * \frac{100}{1} \tag{8.1}$$

This help to visually illustrate the trend in plan quality of both the fixed and the controlled experiment. A decreasing ($\downarrow$) trend in the value of y implies a good quality plan while a continuous increase ($\uparrow$) in value of y means that the planner output is affected by the complexity of the problem in the domain. The more complex the problem becomes the more the challenge to generate quality plan at reasonable time. Moreover, when y is zero, it means the output plan is steady and stable despite an increase in problem complexity.

8.1.2 Evaluation Performance

In this result, we consider the performance of the planner with fixed and controlled signal value. Table 8.1 shows the percentage rate of increase in the queues within the network and the effect of those percentage increase on the average total time; the average number of processes and an average number of actions in the output plans. It is observed that the average total time required to generate a plan varies with a variation in queuing distance and the green split values as shown in Figure 8.2. The percentage change in total time increases with respect to an increase in queue length at fixed signal. However, the percentage change in total time of the controlled signal is remarkable

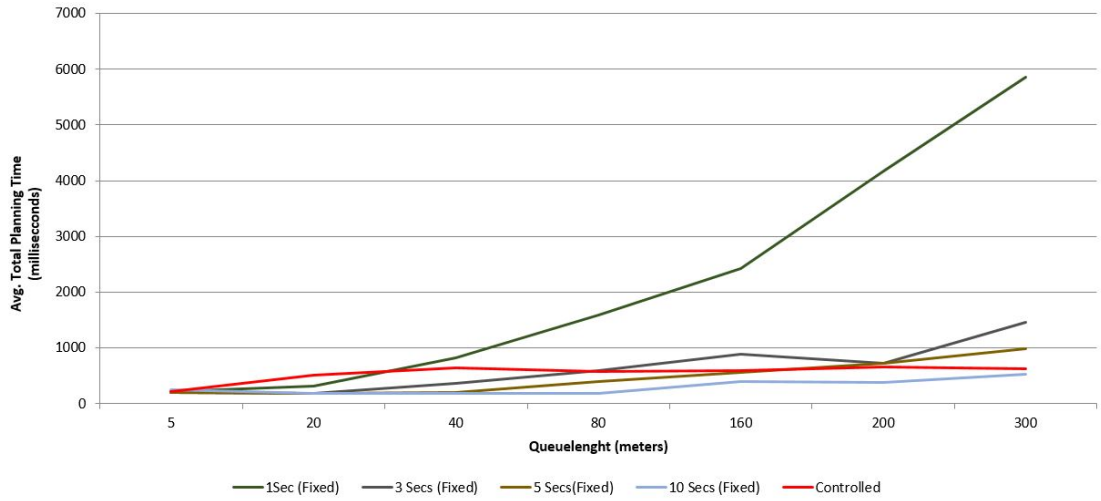

Figure 8.2: The average planning time at different fixed signal timings compared with the controlled UTCPLAN approach with an increasing traffic queues

at a low increase rate with increase in queue length. The result also shows a percentage decrease in the controlled signal at 80 metres and 300 metres queueing distance respectively.

Table 8.1 also shows the percentage change in the average number of processes initiated by the plans. The percentage change in the average number of processes increases with increase in queue length at fixed signal. However, the percentage change in the average number of processes is reduced to zero percent despite an increase in queue length when the signal is controlled by UTCPLAN. It increases a little at 200 metres queueing distance but later drop back to zero percent. It is observed that the total number of initiated process varies with a variation in the green split values as shown in Figure 8.3. The total number of processes initiated by the planner to achieve the goal condition increases with an increase in the congestion rate whenever the signal is fixed. However, the changes are minimum and often becomes steady despite the increasing queues in the network when the green split is controlled by the UTCPLAN approach within the traffic network.

Similarly, Table 8.1 shows the percentage change in the average number of action operator within the plans increases an increase in queue length at fixed signal. However,

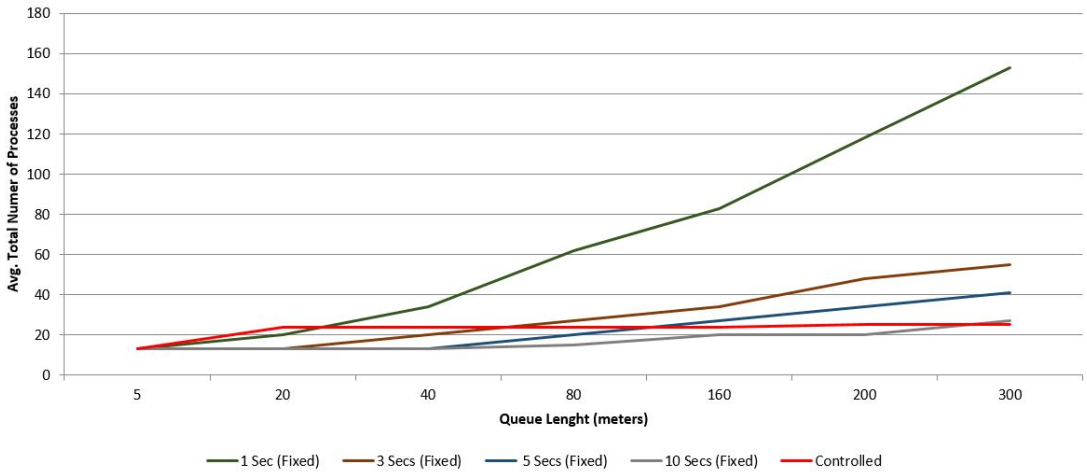

Figure 8.3: The average total number of processes initiated by the planner to achieve the goal condition at different fixed signal timings compared with the controlled UTCPLAN approach with increasing traffic queues

QueueLenght Variation	Increase in QueueLenght(%)	Change in Avg. Planning Time(%)		Change in Avg. No. of Processes(%)		Change in Avg. No.of Actions(%)	
		Fixed	Controlled	Fixed	Controlled	Fixed	Controlled
5	↑ 100	↑ 100	↑ 100	↑ 100	↑ 100	↑ 100	↑ 100
20	↑ 75	↑ 31.3	↑ 56.5	↑ 35.0	↑ 45.8	↑ 50	↑ 58.3
40	↑ 50	↑ 61.2	↑ 22.5	↑ 41.2	0.0	↑ 50	↑ 42.9
80	↑ 50	↑ 48.1	↓ 13.2	↑ 45.2	0.0	↑ 50	0.0
160	↑ 50	↑ 34.5	↑ 3.9	↑ 25.3	0.0	↑ 34.4	0.0
200	↑ 20	↑ 42.1	↑ 8.7	↑ 29.7	↑ 4.0	↑ 29.1	0.0
300	↑ 33.3	↑ 28.5	↓ 5.7	↑ 22.9	0.0	↑ 22.5	0.0

Table 8.1: Planner result showing the percentage increase in number of vehicles in the network and the corresponding percentage changes in the plan generation time; number of processes and actions respectively. Fixed duration means the duration of the green split is fixed at the initial state and would be the same throughout the planning time. Controlled means that the duration is fixed at the initial state, but subject to changes during search space whenever the planner anticipate a better optimised green phase than the fixed value.

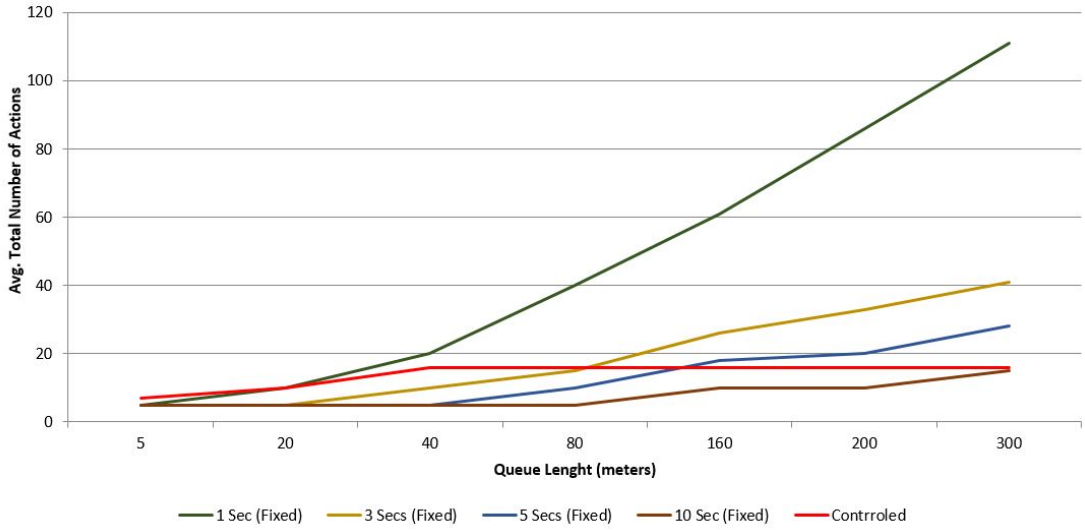

Figure 8.4: The average total number of action operators in the plan at different fixed signal timings compared with the controlled UTCPLAN approach with respect to increasing traffic queues

the percentage change in the average number of action operator is reduced to zero percent despite an increase in queue length when the signal is controlled by UTCPLAN. It is observed that the total number of actions operators that are required to get the problem from initial state to a state that satisfies the goal condition also varies with a variation in the green split values as shown in Figure 8.4. The total number of actions generated by the planner to achieve the goal condition increases with an increase in the traffic congestion rate whenever the signal is fixed. However, the changes are also minimum and often becomes steady despite the increasing queues in the network when the green split is controlled by UTCPLAN approach within the traffic network.

UTCPLAN was also tested on some classical planning problems to see its applicability on other known planning domain. It was able to generate satisficing plans for problems in the "blockworld domain" [96] and "thermal domain" [114] as shown in Appendix H and Appendix G respectively.

8.2 Discussion

The percentage change in output value, as shown in table 8.1, gives a visual illustration of thee trend in plan quality of both the fixed and the controlled experiment. A decreasing ($\downarrow$) trend in the output value implies a good quality plan while a continuous increase ($\uparrow$) in output value means that the planner output is affected by the complexity of the problem in the domain. The more complex the problem becomes, the more the challenge to generate quality plan at reasonable time. Moreover, when the percentage change in output value is zero, it means the output plan is steady and stable despite an increase in problem complexity. Stability in plan metrics can not be achieved by a planner with fixed duration. It can only be achieved by a planner that has the ability to establish a unique approach to numeric fluents during search space. The stability in the controlled output plan metric is achieved through the novel integration of MPC approach with AI planning. This implies that the time to generate a valid plan, as well as the quality of plan generated, becomes stable at some point irrespective of the increase in complexity of the problem domain.

For instance, the result shows that a controlled approach is required to optimise any traffic situation. The ability of the UTCPLAN approach at tracking changes and evaluate the effect of those changes over a period of time in the future helps to anticipate increasing or decreasing queue trends within the network. The controlled green time is always suited to the changes in the network. This help to keep the network in a stable state despite increasing congestion.

The above result shows that both strategies perform well in lesser traffic condition while controlled approach is a little better leading to a reduction of both evaluation criteria. However, there is a vast difference between the two methods when the traffic condition becomes heavier. Controlled approach is superior resulting in a large improvement while the performance of fixed time strategy gets worse. Large traffic demand creates a larger search space and, therefore, the solution requires more computational time espe-

cially at lower fix duration, but with the controlled technique, more substantial demand is not more intensive.

The goal is to minimise vehicles waiting in queue and relieve the congestion in urban traffic networks and the controlled approach provides a good solution. This implies that the average number of vehicles in a queue will almost not increase when intensive flow is added. However, the criteria of fixed time strategy increase significantly which means it cannot adapt to heavy traffic condition and when the signal is fixed for heavy traffic purpose, it cannot adapt to lighter traffic situation.

The planner ability to represent rich representation of the domain makes it easier to take care of some constraints within the road network which a simple MPC controller might not be able to handle. MPC approach, on the other hand, helps to dynamically control the green split within the domain which the searching mechanism might not be able to simulate. Combining the two approaches help to control a signalled junction while still taking care of the logical reasoning within the network of roads.

The planner could be use *online* or *offline*. At the moment, the planner works offline, which means it only generate plans and do not receive any feedback about the execution of the plan nor the state of the environment where the plan is being executed. The generation of plan is only based on the formal description of the environment. The state of the system at the time of executing the plan is assumed to be adequately modelled. Hence, a change in a state prior to plan execution is not taken into consideration. Thus, the domain model has to be robust enough to take care of the gaps or differences between the conceptual model and the real world. An example of offline planning is the generation of a regional advisory plan for traffic operators by a planner to solve an environmental problem using declarative knowledge of traffic strategies and policies within such environment. In this case, the plan generated has to be readable and interpretable by the user for decision making.

However, in a domain where the difference is more than what the planner can handle (in a non-deterministic environment), the planner will need to have constant feedback

from the effect of its actions on the domain environment. Thus, online planning will be employed in this situation. An example of online planning is a real time intelligent traffic controller that uses AI planning for decision making. Such controller will continually sense, interpret and generate execution plans to change its state and the state of the urban traffic situation within its domain of control.

8.2.1 Scaling Difficulties

The AI planning approach in this implementation needs to integrate state-of-the-art heuristics in planning to improve the performance stability and robustness of the planner. Also, good commercial optimisation API need be employed to improve robustness to larger networks of constraints, rather than the simple solver that was employed in this implementation.

8.3 Conclusion

MPC strategy can predict the system behaviours in advance and avoid the unexpected situations with a consideration of constraints explicitly while AI planning approach can reason with both logical and numeric properties of the road network which helps to generate plans during unexpected events or changes within the network. Combining the two approaches is advisable to solve UTC traffic problem.

We tested the implementation of UTCPLAN algorithm to control traffic flow in other to relieve traffic congestion. We introduce congestion to the traffic problem by increasing the percentage of queue length from the source into the network of connected roads. We evaluate the performance of the planner when the traffic is light and when it is congested. Experimental simulations with an objective of minimising the number of vehicles in a queue are implemented to validate the applicability and effectiveness of the algorithm. The result shows that UTCPLAN approach performs well to deal with heavy traffic congestion problem, which might result from heavy traffic flow during

rush hours.

8.4 Summary

This chapter explains the result and evaluation of UTCPLAN algorithm when tested on a network of connected roads. The result shows that UTCPLAN approach has demonstrated to be effective and applicable in urban traffic networks domain.

Chapter 9

Conclusions and Future Work

Most real-world situations involve quantities and numeric values, which are not properly represented in classical planning. The need to express numeric quantities of entities within a domain representation led to the development of metric planning. Also, continuous planning in domains that are represented with rich notations has long been a great challenge for AI. For instance, changes occurring due to fuel consumption, continuous movement, or environmental conditions may not be adequately modelled through instantaneous or even durative actions; rather these require modelling as continuously changing processes. The combination of time-dependent problems and numeric optimisation problem create a more challenging and hard task of time-dependent metric fluents. Creating a continuous planner also requires a real world benchmark, an appropriate benchmark for this kind of problem is an Urban Traffic Control (UTC) problem.

Advanced urban traffic control (UTC) systems are often based on feedback 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 using adaptive green phases and flexible coordination in a 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 need systems that can plan and act effectively in order to restore an unexpected road traffic situation into a normal order. Creating a generic

architecture that enables control systems to reason automatically with knowledge of their environment and their controls, in order to generate plans and schedules to manage themselves, is a significant step forward in the field of Intelligent Transport System.

In this thesis, we describe the design of a generic architecture for UTC which enables the network to manage itself both in normal operation and in unexpected scenarios. To accomplish this, we create a representation that can be used to capture declarative definitions of processes, actions, events, resources and the structure of the environment in a UTC scenario. This description is founded on world states modelled by mixed discrete and continuous state variables. We demonstrate that the description is capable of capturing a wide range of such scenarios, using a realistic example.

The reasoning and self-management aspects are implemented using automated planning techniques inspired by both the symbolic artificial intelligence area and traditional control engineering. Using this technology, we show how the system can reason about unforeseen situations in the road network and generate plans (sequences of actions) that when executed can achieve the desired traffic outcome.

We highlight the challenges of implementing our UTC architecture in the context of continuous on-line planning platform. Significantly, not only do state-of-the-art planning engines fail to find optimal solutions, but their performance significantly decreases on larger road networks.

In the quest to improve reasoning with a continuous process in our architecture and reducing the time taken to generate new plans, we investigate the role of the Model Predictive Control (MPC) approach to planning in the presence of mixed discrete and continuous state variables. Urban traffic control is a flow optimisation problem and is analogous to a certain degree with the problem of controlling chemical flow processes. These kinds of problems are often solved using an MPC strategy. We explore this control approach and show how it can be embedded into existing, modern AI Planning technology. This preserves the many advantages of the AI Planning approach, to

do with domain independence through declarative modelling, and explicit reasoning, while leveraging on the capability of MPC to deal with continuous processes.

Thus, we create a UTC dependent, but scenario and configuration independent planning system called UTCPLAN. The is due to the need to develop planning technology specifically tailored to the parameters of applications such as urban traffic control. We describe the development of UTCPLAN that supports the analysis of domain descriptions and plans containing continuously changing processes, events and actions, and identify new approach for such problems areas which, in particular, takes into account the complexity of the numeric variables involved in urban traffic processes. Such a planner can reason with continuous processes in urban traffic domain and has the potential to generate control and execution plans and schedules that will keep an urban traffic controlled region in a desirable state for the entire life span of traffic flow processes.

We present experimental evaluation showing that our approach can provide plans in a reasonable time. We describe the implementation and performance of our UTCPLAN hybrid algorithm when tested on a network of connected roads. The result shows that UTCPLAN approach has demonstrated to be effective and applicable in urban traffic networks to deal with the congestion problem.

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 divert road traffic from a blocked road without human intervention, thus, satisfying some of the objectives of self-management in transport systems.

9.1 Future Work

We shall integrate state-of-the-art planning heuristics into the search base and plan generations phase of UTCPLAN to reduce search time and optimise makespan during plan

generation. This aspect of our future work would help to improve the performance, stability and scalability of the planner to problems with huge search space.

We also hope to employ good commercial optimisation solver (such as GUROBI) to improve robustness to numeric complexities of larger traffic networks and constraints within the network of connected roads.

This UTCPLAN would also be embedded within our self-management architecture and evaluate its performance on a simulation platform as well as testing it on a real road situation that is an ongoing pilot project in Manchester metropolitan area.

We also plan to provide a deeper evaluation of our approach, especially to compare it with traditional traffic control methods and assess the effort and challenges required to embody the model within a real-world environment. Other aspects of improvement would include:incorporating learning into our architecture which will store plans for the purpose of re-using valid plans and save the cost of re-planing at every instance; incorporating the actual road policies in our model; increase interaction with road users; build a knowledge base from generated (optimal) plans; consider various speed limits; consider weather conditions and priority vehicles.

9.2 Summary

This thesis explains the design, implementation and testing of a UTCPLAN, which can input expressive domain descriptions, output solution plans containing continuous processes, events and actions, and is the first AI Planner we are aware of to integrate MPC with AI search - based planning techniques. This thesis also shows how to embed the knowledge of UTC structures into an augmented AI planning domain. It evaluate the possibility of reasoning with this knowledge to optimise traffic flow in situations where a given road within a network of roads becomes unavailable due to unexpected situations such as road accidents. We show how the problem of self-management of a road traffic network during an unforeseen episode can be solved using an AI planning approach. To solve the continuous planning problem, we specify how to augment the standard AI planning engine with the incorporation of MPC techniques into the central reasoning process. This approach effectively utilises the strengths of search-based and model-simulation-based methods.

Appendix A: A PDDL Domain Model

PDDL file for Urban Traffic Domain model of urban traffic network.

```
(define (domain transport)
  (:requirements :typing :durative-actions :fluents :timed-initial-literals)

  (:types road junction num)

  (:predicates
    (ready ?road - road)
    (operational ?road - road)
    (free ?junction - junction)
    (connected ?road1 - road ?junction - junction ?road2 - road)
  )

  (:functions
    ;(flowrate ?road - road) ; it's not used in the model...
    (capacity ?road - road) ; capacity of the road
    (length ?road - road) ; time taking to drive to the road
    (head ?road - road) ; no of cars at the head of the road
    (tail ?road - road) ; no of cars at the tail of the road
    (use ?road - road) ;number of cars on the road
    (val ?n - num) ;for explicite expressing variable duration, or numbers of cars to drive etc.
   )

  (:durative-action RELEASE-CARS
   :parameters (?r - road)
   :duration (= ?duration 1)
   :condition (and (at start (ready ?r)) (at end (< (use ?r) (capacity ?r))))
   :effect (and (at start (increase (use ?r) 5)) (at start (not (ready ?r))) (at end (ready ?r))(at end (increase (head ?r) 5)))
  )

  (:durative-action DRIVE-THROUGH-JUNCTION
   :parameters (?r1 - road ?j - junction ?r2 - road ?n - num)
   :duration (= ?duration (val ?n))
   :condition (and (at start (>= (tail ?r1) (val ?n))) (over all (connected ?r1 ?j ?r2)) (at start (free ?j)) (over all (operational ?r2)) (at start (<= (+ (us
   :effect (and (at start (not (free ?j)))(at end (free ?j))(at start (decrease (use ?r1) (val ?n)))
      (at start (increase (use ?r2) (val ?n)))(at start (decrease (tail ?r1) (val ?n)))
      (at end (increase (head ?r2) (val ?n))))
  )


  (:durative-action DRIVE
   :parameters (?r - road ?n - num)
```

```
   :duration (= ?duration (length ?r))
   :condition
    (and
      (at start (>= (head ?r) (val ?n)))
      (over all (operational ?r))
    )
   :effect
    (and
(at start (decrease (head ?r) (val ?n)))
(at end (increase (tail ?r) (val ?n)))
;(at end (increase (tail ?r) 5))
;(at start (assign (head ?r) 0))

    )
  )
;  (:durative-action FLOW2
;   :parameters (?road1 ?road2 - road)
;   :duration (= ?duration 1)
;   :condition
;    (and
;      (at start (connected ?road1 ?road2))
;      (at start (>= (capacity ?road2) (+ (use ?road2) (flowrate ?road1))))
;      (at start (< (use ?road1) (flowrate ?road1)))
;      (at start (> (use ?road1) 0))
;      (at start (operational ?road2))
;    )
;   :effect
;    (and
;      (at end (decrease (use ?road1) (use ?road1)))
;      (at end (increase (use ?road2) (use ?road1)))
;    )
;  )

)
```

Appendix B: PDDL Problem File for an Urban Traffic Network

PDDL Problem File for the Urban Traffic Network area of Huddersfield Town center area of West Yorkshire in United Kingdom tested on some domain independent planners. Each of the alphabets represents corresponding names of objects in the network.

```
(define (problem hud1)
 (:domain transport)
 (:objects
a1 a2 b c1 c2 d1 d2 e1 e2 f1 f2 g1 g2 h i j k l m n1 n2 o p q r s t u v1 v2 w x y1 y2 z zz - road
j1 j2 j3 j4 j5 j6 j7 j8 j9 j10 j11 j12 j13 - junction
        nn1 nn2 nn3 nn4 nn5 - num
 )
 (:init
   (connected a1 j1 g2)
   (connected a1 j1 c2)
   (connected c1 j1 a2)
   (connected c1 j1 g2)
   (connected g1 j1 c2)
   (connected g1 j1 a2)

   (connected c2 j2 b)
   (connected c2 j2 d2)
   (connected zz j2 c1)
   (connected zz j2 d2)
   (connected zz j2 b)
   (connected d1 j2 c1)
   (connected d1 j2 b)

   (connected d2 j3 e2)
   (connected e1 j3 d1)
   (connected z j3 d1)
   (connected z j3 e2)

   (connected e2 j4 f2)
   (connected e2 j4 y1)
   (connected f1 j4 e1)
   (connected f1 j4 y1)
   (connected y2 j4 e1)
   (connected y2 j4 f2)

   (connected g2 j5 m)
```

```
(connected i j5 g1)
(connected i j5 m)

(connected h j6 i)
(connected h j6 j)

(connected j j7 k)
(connected l j7 k)

(connected m j8 l)
(connected m j8 n1)
(connected t j8 l)
(connected t j8 n1)
(connected n2 j8 l)

(connected n1 j9 q)
(connected n1 j9 o)

(connected u j10 t)
(connected u j10 s)
(connected u j10 zz)

(connected q j11 r)
(connected q j11 p)
(connected s j11 r)
(connected s j11 p)

(connected r j12 u)
(connected r j12 z)
(connected r j12 v1)
(connected v2 j12 u)
(connected v2 j12 z)

(connected v1 j13 y2)
(connected v1 j13 x)
(connected y1 j13 v2)
(connected y1 j13 x)
(connected w j13 v2)
(connected w j13 y2)
(connected w j13 x)


(free j1)
(free j2)
(free j3)
(free j4)
(free j5)
(free j6)
(free j7)
(free j8)
(free j9)
(free j10)
(free j11)
(free j12)
(free j13)

(= (capacity a1) 10)
```

```
(= (capacity a2) 10)
(= (capacity b) 10)
(= (capacity c1) 4)
(= (capacity c2) 4)
(= (capacity d1) 3)
(= (capacity d2) 3)
(= (capacity e1) 5)
(= (capacity e2) 5)
(= (capacity f1) 5)
(= (capacity f2) 5)
(= (capacity g1) 4)
(= (capacity g2) 4)
(= (capacity h) 3)
(= (capacity i) 4)
(= (capacity j) 3)
(= (capacity k) 10)
(= (capacity l) 4)
(= (capacity m) 3)
(= (capacity n1) 5)
(= (capacity n2) 5)
(= (capacity o) 5)
(= (capacity p) 5)
(= (capacity q) 3)
(= (capacity r) 7)
(= (capacity s) 5)
(= (capacity t) 3)
(= (capacity u) 3)
(= (capacity v1) 5)
(= (capacity v2) 5)
(= (capacity w) 10)
(= (capacity x) 4)
(= (capacity y1) 7)
(= (capacity y2) 7)
(= (capacity z) 7)
(= (capacity zz) 7)

 (= (length a1) 10)
(= (length a2) 10)
(= (length b) 10)
(= (length c1) 4)
(= (length c2) 4)
(= (length d1) 3)
(= (length d2) 3)
(= (length e1) 5)
(= (length e2) 5)
(= (length f1) 5)
(= (length f2) 5)
(= (length g1) 4)
(= (length g2) 4)
(= (length h) 3)
(= (length i) 4)
(= (length j) 3)
(= (length k) 10)
(= (length l) 4)
(= (length m) 3)
(= (length n1) 5)
(= (length n2) 5)
```

```
   (= (length o) 5)
   (= (length p) 5)
   (= (length q) 3)
   (= (length r) 7)
   (= (length s) 5)
   (= (length t) 3)
   (= (length u) 3)
   (= (length v1) 5)
   (= (length v2) 5)
   (= (length w) 10)
   (= (length x) 4)
   (= (length y1) 7)
   (= (length y2) 7)
   (= (length z) 7)
   (= (length zz) 7)

(= (use a1) 0)
   (= (use a2) 0)
   (= (use b) 0)
   (= (use c1) 0)
   (= (use c2) 0)
   (= (use d1) 0)
   (= (use d2) 0)
   (= (use e1) 0)
   (= (use e2) 0)
   (= (use f1) 0)
   (= (use f2) 0)
   (= (use g1) 0)
   (= (use g2) 0)
   (= (use h) 0)
   (= (use i) 0)
   (= (use j) 0)
   (= (use k) 0)
   (= (use l) 0)
   (= (use m) 0)
   (= (use n1) 0)
   (= (use n2) 0)
   (= (use o) 0)
   (= (use p) 0)
   (= (use q) 0)
   (= (use r) 0)
   (= (use s) 0)
   (= (use t) 0)
   (= (use u) 0)
   (= (use v1) 0)
   (= (use v2) 0)
   (= (use w) 0)
   (= (use x) 0)
   (= (use y1) 0)
   (= (use y2) 0)
   (= (use z) 0)
   (= (use zz) 0)

(= (head a1) 0)
   (= (head a2) 0)
   (= (head b) 0)
   (= (head c1) 0)
```

```
   (= (head c2) 0)
   (= (head d1) 0)
   (= (head d2) 0)
   (= (head e1) 0)
   (= (head e2) 0)
   (= (head f1) 0)
   (= (head f2) 0)
   (= (head g1) 0)
   (= (head g2) 0)
   (= (head h) 0)
   (= (head i) 0)
   (= (head j) 0)
   (= (head k) 0)
   (= (head l) 0)
   (= (head m) 0)
   (= (head n1) 0)
   (= (head n2) 0)
   (= (head o) 0)
   (= (head p) 0)
   (= (head q) 0)
   (= (head r) 0)
   (= (head s) 0)
   (= (head t) 0)
   (= (head u) 0)
   (= (head v1) 0)
   (= (head v2) 0)
   (= (head w) 0)
   (= (head x) 0)
   (= (head y1) 0)
   (= (head y2) 0)
   (= (head z) 0)
   (= (head zz) 0)

(= (tail a1) 0)
   (= (tail a2) 0)
   (= (tail b) 0)
   (= (tail c1) 0)
   (= (tail c2) 0)
   (= (tail d1) 0)
   (= (tail d2) 0)
   (= (tail e1) 0)
   (= (tail e2) 0)
   (= (tail f1) 0)
   (= (tail f2) 0)
   (= (tail g1) 0)
   (= (tail g2) 0)
   (= (tail h) 0)
   (= (tail i) 0)
   (= (tail j) 0)
   (= (tail k) 0)
   (= (tail l) 0)
   (= (tail m) 0)
   (= (tail n1) 0)
   (= (tail n2) 0)
   (= (tail o) 0)
   (= (tail p) 0)
   (= (tail q) 0)
```

```
(= (tail r) 0)
(= (tail s) 0)
(= (tail t) 0)
(= (tail u) 0)
(= (tail v1) 0)
(= (tail v2) 0)
(= (tail w) 0)
(= (tail x) 0)
(= (tail y1) 0)
(= (tail y2) 0)
(= (tail z) 0)
(= (tail zz) 0)


(= (val nn1) 1)
(= (val nn2) 2)
(= (val nn3) 3)
(= (val nn4) 4)
(= (val nn5) 5)

(operational a1)
(operational a2)
(operational b)
(operational c1)
(operational c2)
(operational d1)
(operational d2)
(operational e1)
(operational e2)
(operational f1)
(operational f2)
(operational g1)
(operational g2)
(operational h)
(operational i)
(operational j)
(operational k)
(operational l)
(operational m)
(operational n1)
(operational n2)
(operational o)
(operational p)
(operational q)
(operational r)
(operational s)
(operational t)
(operational u)
(operational v1)
(operational v2)
(operational w)
(operational x)
(operational y1)
(operational y2)
(operational z)
(operational zz)
```

```
;   (ready a)
;   (at 0.1 (not (ready a)))
;   (at 10.0 (ready a))
;   (at 10.1 (not (ready a)))
;   (at 20.0 (ready a))
;   (at 20.1 (not (ready a)))
;   (at 30.0 (ready a))
;   (at 30.1 (not (ready a)))
;   (at 40.0 (ready a))
;   (at 40.1 (not (ready a)))
;   (at 50.0 (ready a))
;   (at 50.1 (not (ready a)))

   (ready w)
;   (at 0.1 (not (ready e)))
;   (at 10.0 (ready e))
;   (at 10.1 (not (ready e)))
;   (at 20.0 (ready e))
;   (at 20.1 (not (ready e)))
;   (at 30.0 (ready e))
;   (at 30.1 (not (ready e)))
;   (at 40.0 (ready e))
;   (at 40.1 (not (ready e)))
;   (at 40.0 (ready e))
;   (at 40.1 (not (ready e)))
 )
 (:goal
   (and
    ;(>= (tail a) 5)
    ;(>= (tail j) 2)
    ;(>= (tail k) 0)
    (>= (tail b) 5)
   )
 )
 (:metric minimize (total-time))
)
```

Appendix C: Sample Plan Generated by a Domain Independent Planner

The plan generated by the Optic planner for Problem 1 of PDDL Problem File for the Urban Traffic Network area of Huddersfield Town center area of West Yorkshire in United Kingdom when tested on some domain independent planners. Each of the alphabets represents corresponding names of objects in the network without considering blockages

```
0.000: (release-cars w)  [1.000]
1.001: (drive w nn1)  [10.000]
1.002: (drive w nn1)  [10.000]
1.003: (drive w nn1)  [10.000]
1.004: (drive w nn1)  [10.000]
1.005: (drive w nn1)  [10.000]
11.002: (drive-through-junction w j13 y2 nn1)  [1.000]
12.003: (drive y2 nn1)  [7.000]
12.003: (drive-through-junction w j13 v2 nn1)  [1.000]
13.004: (drive v2 nn1)  [5.000]
18.005: (drive-through-junction v2 j12 u nn1)  [1.000]
18.006: (drive-through-junction w j13 v2 nn1)  [1.000]
19.004: (drive-through-junction y2 j4 e1 nn1)  [1.000]
19.006: (drive u nn1)  [3.000]
19.007: (drive v2 nn1)  [5.000]
19.007: (drive-through-junction w j13 y2 nn1)  [1.000]
20.005: (drive e1 nn1)  [5.000]
20.008: (drive y2 nn1)  [7.000]
22.007: (drive-through-junction u j10 zz nn1)  [1.000]
23.008: (drive zz nn1)  [7.000]
24.008: (drive-through-junction v2 j12 z nn1)  [1.000]
24.009: (drive-through-junction w j13 v2 nn1)  [1.000]
25.006: (drive-through-junction e1 j3 d1 nn1)  [1.000]
25.009: (drive z nn1)  [7.000]
25.010: (drive v2 nn1)  [5.000]
26.007: (drive d1 nn1)  [3.000]
27.009: (drive-through-junction y2 j4 e1 nn1)  [1.000]
28.010: (drive e1 nn1)  [5.000]
```

```
29.008: (drive-through-junction d1 j2 b nn1)  [1.000]
30.009: (drive b nn1)  [10.000]
30.009: (drive-through-junction zz j2 b nn1)  [1.000]
30.011: (drive-through-junction v2 j12 u nn1)  [1.000]
31.010: (drive b nn1)  [10.000]
31.012: (drive u nn1)  [3.000]
32.010: (drive-through-junction z j3 d1 nn1)  [1.000]
33.011: (drive d1 nn1)  [3.000]
34.013: (drive-through-junction u j10 zz nn1)  [1.000]
35.014: (drive zz nn1)  [7.000]
36.012: (drive-through-junction d1 j2 b nn1)  [1.000]
36.013: (drive-through-junction e1 j3 d1 nn1)  [1.000]
37.013: (drive b nn1)  [10.000]
37.014: (drive d1 nn1)  [3.000]
40.015: (drive-through-junction d1 j2 b nn1)  [1.000]
41.016: (drive b nn1)  [10.000]
42.015: (drive-through-junction zz j2 b nn1)  [1.000]
43.016: (drive b nn1)  [10.000]
}
```

Appendix D: Sample plan generated by UTCPLAN

The plan generated by UTCPLAN at a fix duration of 5seconds count when both phase are active and no blockage in any of the connected roads.

```
****************************************************************************************
Plan . . . .



MAKESPAN OPERATOR STATUS PARAMETER DURATION
0.0 switch_to_green [JunctionA, phase1, nLNorth, nLSouth] 5.0
5.0 switch_to_green [JunctionA, phase2, jWNorth, jWSouth] 5.0
10.0 switch_to_green [JunctionB, phase2, brStr, lDSouth] 5.0
15.0 [process] Jtraffic_flow [JunctionA, phase1, nLNorth, nLSouth] 5.0
20.0 [process] Jtraffic_flow [JunctionA, phase2, jWNorth, jWSouth] 5.0
25.0 [process] Jtraffic_flow [JunctionB, phase2, brStr, lDSouth] 5.0
40.0 switch_to_green [JunctionA, phase1, nLNorth, nLSouth] 5.0
45.0 switch_to_green [JunctionA, phase2, jWNorth, jWSouth] 5.0
50.0 switch_to_green [JunctionB, phase2, brStr, lDSouth] 5.0
55.0 [process] Jtraffic_flow [JunctionA, phase1, nLNorth, nLSouth] 5.0
60.0 [process] Jtraffic_flow [JunctionA, phase2, jWNorth, jWSouth] 5.0
65.0 [process] Jtraffic_flow [JunctionB, phase2, brStr, lDSouth] 5.0
70.0 [process] Rtraffic_flow [nLSouth, wDStr] 5.0
75.0 [process] Rtraffic_flow [lDSouth, pTStr] 5.0
90.0 switch_to_green [JunctionA, phase1, nLNorth, nLSouth] 5.0
95.0 switch_to_green [JunctionA, phase2, jWNorth, jWSouth] 5.0
100.0 switch_to_green [JunctionB, phase1, wDStr, lDSouth] 5.0
105.0 switch_to_green [JunctionB, phase2, brStr, lDSouth] 5.0
110.0 switch_to_green [JunctionC, phase1, pTStr, bmStr] 5.0
115.0 [process] Jtraffic_flow [JunctionA, phase1, nLNorth, nLSouth] 5.0
120.0 [process] Jtraffic_flow [JunctionA, phase2, jWNorth, jWSouth] 5.0
125.0 [process] Jtraffic_flow [JunctionB, phase1, wDStr, lDSouth] 5.0
130.0 [process] Jtraffic_flow [JunctionB, phase2, brStr, lDSouth] 5.0
135.0 [process] Jtraffic_flow [JunctionC, phase1, pTStr, bmStr] 5.0
140.0 [process] Rtraffic_flow [nLSouth, wDStr] 5.0
145.0 [process] Rtraffic_flow [lDSouth, pTStr] 5.0
160.0 switch_to_green [JunctionA, phase1, nLNorth, nLSouth] 5.0
```

```
165.0 switch_to_green [JunctionA, phase2, jWNorth, jWSouth] 5.0
170.0 switch_to_green [JunctionB, phase1, wDStr, lDSouth] 5.0
175.0 switch_to_green [JunctionB, phase2, brStr, lDSouth] 5.0
180.0 switch_to_green [JunctionC, phase1, pTStr, bmStr] 5.0
185.0 [process] Jtraffic_flow [JunctionA, phase1, nLNorth, nLSouth] 5.0
190.0 [process] Jtraffic_flow [JunctionA, phase2, jWNorth, jWSouth] 5.0
195.0 [process] Jtraffic_flow [JunctionB, phase1, wDStr, lDSouth] 5.0
200.0 [process] Jtraffic_flow [JunctionB, phase2, brStr, lDSouth] 5.0
205.0 [process] Jtraffic_flow [JunctionC, phase1, pTStr, bmStr] 5.0
210.0 [process] Rtraffic_flow [nLSouth, wDStr] 5.0
215.0 [process] Rtraffic_flow [lDSouth, pTStr] 5.0
230.0 switch_to_green [JunctionA, phase1, nLNorth, nLSouth] 5.0
235.0 switch_to_green [JunctionA, phase2, jWNorth, jWSouth] 5.0
240.0 switch_to_green [JunctionB, phase1, wDStr, lDSouth] 5.0
245.0 switch_to_green [JunctionB, phase2, brStr, lDSouth] 5.0
250.0 switch_to_green [JunctionC, phase1, pTStr, bmStr] 5.0
255.0 [process] Jtraffic_flow [JunctionA, phase1, nLNorth, nLSouth] 5.0
260.0 [process] Jtraffic_flow [JunctionA, phase2, jWNorth, jWSouth] 5.0
265.0 [process] Jtraffic_flow [JunctionB, phase1, wDStr, lDSouth] 5.0
270.0 [process] Jtraffic_flow [JunctionB, phase2, brStr, lDSouth] 5.0
275.0 [process] Jtraffic_flow [JunctionC, phase1, pTStr, bmStr] 5.0
280.0 [process] Rtraffic_flow [nLSouth, wDStr] 5.0
285.0 [process] Rtraffic_flow [lDSouth, pTStr] 5.0
300.0 switch_to_green [JunctionA, phase1, nLNorth, nLSouth] 5.0
305.0 switch_to_green [JunctionA, phase2, jWNorth, jWSouth] 5.0
310.0 switch_to_green [JunctionB, phase1, wDStr, lDSouth] 5.0
315.0 switch_to_green [JunctionB, phase2, brStr, lDSouth] 5.0
320.0 switch_to_green [JunctionC, phase1, pTStr, bmStr] 5.0
325.0 [process] Jtraffic_flow [JunctionA, phase1, nLNorth, nLSouth] 5.0
330.0 [process] Jtraffic_flow [JunctionA, phase2, jWNorth, jWSouth] 5.0
335.0 [process] Jtraffic_flow [JunctionB, phase1, wDStr, lDSouth] 5.0
340.0 [process] Jtraffic_flow [JunctionB, phase2, brStr, lDSouth] 5.0
345.0 [process] Jtraffic_flow [JunctionC, phase1, pTStr, bmStr] 5.0
350.0 [process] Rtraffic_flow [nLSouth, wDStr] 5.0
355.0 [process] Rtraffic_flow [lDSouth, pTStr] 5.0
370.0 switch_to_green [JunctionB, phase1, wDStr, lDSouth] 5.0
375.0 switch_to_green [JunctionC, phase1, pTStr, bmStr] 5.0
380.0 [process] Jtraffic_flow [JunctionA, phase1, nLNorth, nLSouth] 5.0
385.0 [process] Jtraffic_flow [JunctionA, phase2, jWNorth, jWSouth] 5.0
390.0 [process] Jtraffic_flow [JunctionB, phase1, wDStr, lDSouth] 5.0
395.0 [process] Jtraffic_flow [JunctionB, phase2, brStr, lDSouth] 5.0
400.0 [process] Jtraffic_flow [JunctionC, phase1, pTStr, bmStr] 5.0



Total Makespan:  405.0
Numebr of Actions Operators in PLAN:  28
Numebr of executed PROCESSES in PLAN:  41
Total Output Steps:  81


Node Count:  14


Timings:


Preprocessing Time is   58
```

```
Total Searching Time is   1278
Total Planning Time is   1336
Total Planning Time in Seconds:   ~ 1seconds
```

Appendix E : Sample UTCPLAN Domain Model

Road Traffic Domain Model in within UTCPLAN showing the java declaration of the domain and the formatted output representation of the domain.

```
public TrafficDomain(){
      this.dName = ("Traffic Domain");


    // this.t.setTime(10);
  }// construct



//DECLARE ALL PREDICATES
   public  Predicate intersect(Junction j1,
                   Phase p1,
                   Road r1, Road r2)//predicate intersect(Road1, Road2)
    {
        //String intersect = new String();
          return getPredicate("intersect",
                   j1.junctionName, p1.phaseName,
                   r1.roadName, r2.roadName);
    }


    public  Predicate link(Road r1, Road r2)//predicate intersect(Road1, Road2)
    {
```

```
      //String intersect = new String();
        return getPredicate("link", r1.roadName, r2.roadName);
  }

  public  Predicate JflowActive(Junction j1,
  Phase p1, Road r1, Road r2)
  //predicate intersect(Road1, Road2)
  {
      //String intersect = new String();
        return getPredicate("JflowActive",
        j1.junctionName, p1.phaseName, r1.roadName, r2.roadName);
  }

   public  Predicate RflowActive(Road r1, Road r2)//predicate intersect(Road1, Road2)
  {
      //String intersect = new String();
        return getPredicate("RflowActive", r1.roadName, r2.roadName);
  }


//DECLARE ALL FUNCTIONS
  public  CFunction queue(String operand,
  Road r1, float value)//function queuelenght(Road1)
 {    return getFunction("queueLenght",
 operand, r1, value, cfunctionList);

 }

   public  CFunction queue(String operand,
   Road r1, NValues nvalue)
   //function queuelenght(Road1)
 {    return getFunction("queueLenght",
          operand, r1, nvalue, cfunctionList);
```

```
  }

  public  CFunction capacity(String operand, Road r1, float value )
   {
         CFunction capacity = new CFunction();
         capacity.setTerminology("capacity");
         capacity.setcFunction(operand, r1.roadName, value);
         cfunctionList.add(capacity);
         return capacity;
   }

  //DECLARE ALL FUNCTIONS
   public  CFunction interupt(String operand,
               Road r1, float value)//function queuelenght(Road1)
  {    return getFunction("interuptLevel",
                    operand, r1, value, cfunctionList);

  }

    public  CFunction interupt(String operand,
               Road r1, NValues nvalue)//function queuelenght(Road1)
  {    return getFunction("interuptLevel", operand, r1, nvalue, cfunctionList);
          //cfunctionList.add(queue);
        //return queue;
  }


//DECLARE ALL ACTION OPERATORS>>>>>>>
//Action greenSplit for Road1 and Road2 at Junction 1
       public Action greenSplit(Junction j1, Phase p1, Road r1, Road r2)
   {   declareSplitAction(j1, p1, r1,r2);
       Action switch_to_green = new Action();
```

```
        switch_to_green.setDuration(adur);
       switch_to_green.setAction("switch_to_green",
                j1.junctionName, p1.phaseName,
                r1.roadName, r2.roadName);
       switch_to_green.setPreconditions(intersect(j1, p1, r1, r2),
       queue(">", r1, 0) ,
       interupt("<", r2, 7));
       switch_to_green.setEffects(JflowActive(j1, p1, r1, r2));
      //switch_to_green.setcEffects(queue("assign", r1, 0),
      queue("assign", r2, 0));//increase roadA decrease roadB
     actionList.add(switch_to_green); //add action to list of actions
           opList.add(switch_to_green); //add action to list of actions
           return switch_to_green;
   }

       //DECLARATION for greenSplit for Road1 and Road2 at Junction 1
        void declareSplitAction(Junction j1, Phase p1, Road r1, Road r2)
   {
       Action switch_to_green = new Action();
        switch_to_green.setAction("switch_to_green",
                   j1.junctionName, p1.phaseName,
                       r1.roadName, r2.roadName);
        switch_to_green.setDuration(adur);
       switch_to_green.setPreconditions(setIntersect(j1, p1, r1, r2),
                queueLenght(">", r1, 0),
                interuptLevel("<", r2, 7));
       switch_to_green.setEffects(setJFlowActive(j1,p1, r1, r2));
     // switch_to_green.setcEffects(queueLenght("assign", r1, 0),
     queueLenght("assign", r2, 0));//increase roadA decrease roadB

   }

//DECLARE GROUNDED PROCESSES>>>>>>
```

```
// PROCESS traffic_flow for Road1 and Road2 at Junction 1
        public Proces JtrafficFlow(Junction j1 , Phase p1, Road r1, Road r2)
    {   InitNValues();
        declaretrafficFlow(j1, p1, r1,r2);
        Proces Jtraffic_flow = new Proces();
                Jtraffic_flow.setProces("Jtraffic_flow",
                    j1.junctionName,  p1.phaseName,
                         r1.roadName, r2.roadName);
         Jtraffic_flow.setDuration(adur);
       // traffic_flow.setPreconditions(intersect(j1, r1, r2),queue(">=", r1, 10));
        Jtraffic_flow.setPreconditions(JflowActive(j1,p1, r1, r2));
       Jtraffic_flow.setcEffects(queue("decrease", r1, calJVehicleNo),
       queue("increase", r2, calJVehicleNo));//increase roadA decrease roadB

            prcList.add(Jtraffic_flow); //add process to process List
        opList.add(Jtraffic_flow); //add process to operator list List
            return Jtraffic_flow;
    }

        //DECLARATION for traffic_flow for Road1 and Road2 at Junction 1
         void declaretrafficFlow(Junction j1 , Phase p1, Road r1, Road r2)
    {   InitNValues();
        Proces Jtraffic_flow = new Proces();
         Jtraffic_flow.setProces("Jtraffic_flow", j1.junctionName,
         p1.phaseName, r1.roadName, r2.roadName);
        Jtraffic_flow.setDuration(adur);
        //traffic_flow.setPreconditions(setIntersect(j1, r1, r2),
        queueLenght(">=", r1, 10));
        Jtraffic_flow.setPreconditions(setJFlowActive(j1, p1, r1, r2));
       Jtraffic_flow.setcEffects(queueLenght("decrease", r1, calJVehicleNo),
       queueLenght("increase", r2, calJVehicleNo));//increase roadA decrease roadB
    }
```

```
//DECLARE GROUNDED PROCESSES>>>>>>>>>>>
// PROCESS traffic_flow for Road1 and Road2 at Junction 1
        public Proces RtrafficFlow(Road r1, Road r2)
    {   InitNValues();
        declareRtrafficFlow(r1,r2);
        Proces Rtraffic_flow = new Proces();
        Rtraffic_flow.setProces("Rtraffic_flow", r1.roadName, r2.roadName);
         Rtraffic_flow.setDuration(adur);
       // traffic_flow.setPreconditions(intersect(j1, r1, r2),queue(">=", r1, 10));
        Rtraffic_flow.setPreconditions(RflowActive(r1, r2));
       Rtraffic_flow.setcEffects(queue("decrease", r1, calJVehicleNo),
       queue("increase", r2, calJVehicleNo));//increase roadA decrease roadB

            prcList.add(Rtraffic_flow); //add process to process List
        opList.add(Rtraffic_flow); //add process to operator list List
            return Rtraffic_flow;
    }

        //DECLARATION for traffic_flow for Road1 and Road2 at Junction 1
         void declareRtrafficFlow(Road r1, Road r2)
    {   InitNValues();
        Proces Rtraffic_flow = new Proces();
        Rtraffic_flow.setProces("Rtraffic_flow", r1.roadName, r2.roadName);
        Rtraffic_flow.setDuration(adur);
        //traffic_flow.setPreconditions(setIntersect(j1, r1, r2),
        queueLenght(">=", r1, 10));
        Rtraffic_flow.setPreconditions(setRFlowActive(r1, r2));
       Rtraffic_flow.setcEffects(queueLenght("decrease", r1, calJVehicleNo),
       queueLenght("increase", r2, calJVehicleNo));
       //increase roadA decrease roadB
    }
```

```
 //DECLARE GROUNDED EVENTS>>>>>>>>>>
//external events in the domain checking status of road
        public Eventt linkingRoads(Road r1, Road r2)
    {   declarelinkingRoads(r1,r2);
        Eventt road_linked = new Eventt();
        road_linked.setEvent("road_linked", r1.roadName, r2.roadName);
         road_linked.setDuration(adur);
        road_linked.setPreconditions(link(r1, r2),
                queue(">", r1, 0) , interupt("<", r2, 7));
        road_linked.setEffects(RflowActive(r1, r2));
       evtList.add(road_linked); //add action to list of actions
            opList.add(road_linked); //add action to list of actions
            return road_linked;
    }

        //DECLARATION for greenSplit for Road1 and Road2 at Junction 1
         void declarelinkingRoads(Road r1, Road r2)
    {
       Eventt road_linked = new Eventt();
        road_linked.setEvent("road_linked", r1.roadName, r2.roadName);
        road_linked.setDuration(adur);
        road_linked.setPreconditions(setLink(r1, r2),
                queueLenght(">", r1, 0) , interuptLevel("<", r2, 7));
        road_linked.setEffects(RflowActive(r1, r2));

    }


//internal event checking status of road
        public Eventt upstreamFilled(Junction j1, Road r1, Road r2)
    {   InitNValues();
        declareUpstreamFilled(j1, r1,r2);
        Eventt upstreamFilled = new Eventt();
```

```
        upstreamFilled.setEvent("upstreamFilled",
                j1.junctionName, r1.roadName, r2.roadName);
         upstreamFilled.setDuration(1);
       // traffic_flow.setPreconditions(intersect(j1, r1, r2),
            queue(">=", r1, 10));
        upstreamFilled.setPreconditions(queue(">=", r2, 200));
       upstreamFilled.setcEffects(queue("assign", r2, 200),
                interupt("assign", r2, 7));//create an interupt

            evtList.add(upstreamFilled); //add process to process List
        opList.add(upstreamFilled); //add process to operator list List
            return upstreamFilled;
    }

        //DECLARATION for traffic_flow for Road1 and Road2 at Junction 1
         void declareUpstreamFilled(Junction j1, Road r1, Road r2)
    {   InitNValues();
       Eventt upstreamFilled = new Eventt();
        upstreamFilled.setEvent("upstreamFilled", j1.junctionName,
                r1.roadName, r2.roadName);
         upstreamFilled.setDuration(1);
        //traffic_flow.setPreconditions(setIntersect(j1, r1, r2),
                queueLenght(">=", r1, 10));
        upstreamFilled.setPreconditions(queueLenght(">=", r2, 200));
       upstreamFilled.setcEffects(queueLenght("assign", r2, 200),
                interuptLevel("assign", r2, 7));//increase roadA decrease roadB
    }


//DECLARE ALL CLACULATIONS AND TRAFFIC FORMULARS

// initialise all numeric formular for calculating dynamic continous changes
    public  void InitNValues(){
```

```
 calJVehicleNo.setNValues("*",CONSTANTS.Fj ,
                jActiveTime.getTime());
//junction flowrate * active green time = number of vehicles leaving J
calRVehicleNo.setNValues("*",CONSTANTS.Fz,
         rActiveTime.getTime());
         //road flowrate * road active time=number of vehicles leaving R
calJTime.setNValues("*", 10,1);

}



//DECLARE MICELLENIOUS UTILITY METHODS>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>

public  Predicate setIntersect(Junction j1, Phase p1, Road r1, Road r2)
{
      Predicate intersect = new Predicate();
  intersect.setPredicate("intersect",
                        j1.junctionName, p1.phaseName,
                         r1.roadName,r2.roadName);
      predicateList.add(intersect);
      return intersect;
}

public  Predicate setLink(Road r1, Road r2)
{
      Predicate link = new Predicate();
  link.setPredicate("link", r1.roadName,r2.roadName);
      predicateList.add(link);
      return link;
}

public  Predicate setJFlowActive(Junction j1, Phase p1, Road r1, Road r2)
```

```
  {
        Predicate JflowActive = new Predicate();
    JflowActive.setPredicate("JflowActive",
                        j1.junctionName, p1.phaseName,
                        r1.roadName,r2.roadName);
        predicateList.add(JflowActive);
        return JflowActive;
  }

 public  Predicate setRFlowActive(Road r1, Road r2)
  {
        Predicate RflowActive = new Predicate();
    RflowActive.setPredicate("RflowActive", r1.roadName,r2.roadName);
        predicateList.add(RflowActive);
        return RflowActive;
  }

 public  Predicate setStatus(Road r1, Road r2)
  {
        Predicate status = new Predicate();
    status.setPredicate("status", r1.roadName,r2.roadName);
        predicateList.add(status);
        return status;
  }



public  Predicate status(String status , Road r1, Road r2)
  {
      //String intersect = new String();
        return getPredicate("status", r1.roadName, r2.roadName);
  }
```

```
public Predicate getPredicate(String name, String obj1, String obj2 ) {
    Predicate  pred = new Predicate();

      for(int index= 0;index<predicateList.size();index++ ){

        Predicate  p = predicateList.get(index);
      //if ((p.getName() == "intersect")
                  && (p.getObj(0) == obj1)
                  && (p.getObj(1) == obj1))
        //  return p;}
 if ((p.getName().equals(name))
          && (p.getObj(0).equals(obj1))
          && (p.getObj(1).equals(obj2)))
         pred = p;
      }
              return pred;
}


 public Predicate getPredicate(String name,
                          String obj1, String obj2,
                                  String obj3 , String obj4 ) {
      Predicate  pred = new Predicate();

      for(int index= 0;index<predicateList.size();index++ ){

        Predicate  p = predicateList.get(index);
      //if ((p.getName() == "intersect")
      && (p.getObj(0) == obj1)
      && (p.getObj(1) == obj1))
        //  return p;}
 if ((p.getName().equals(name))
         && (p.getObj(0).equals(obj1))
```

```
              && (p.getObj(1).equals(obj2))
                  && (p.getObj(2).equals(obj3))
                      && (p.getObj(3).equals(obj4)))
         pred = p;
      }
              return pred;
}


 public Predicate getPredicate(String name, String obj1, String obj2, String obj3 ) {
      Predicate  pred = new Predicate();

      for(int index= 0;index<predicateList.size();index++ ){

        Predicate  p = predicateList.get(index);
      //if ((p.getName() == "intersect")
      && (p.getObj(0) == obj1)
      && (p.getObj(1) == obj1))
        //  return p;}
 if ((p.getName().equals(name))
         && (p.getObj(0).equals(obj1))
              && (p.getObj(1).equals(obj2))
                  && (p.getObj(2).equals(obj3)))
         pred = p;
      }
              return pred;
}
```

```
    public  CFunction queueLenght(String operand, Road r1, float value)
   {   CFunction queue = new CFunction();
           queue.getFunction("queueLenght", operand, r1.roadName, value);
           cfunctionList.add(queue);
         return queue;
    }

      public  CFunction queueLenght(String operand, Road r1, NValues nvalue)
   {   CFunction queue = new CFunction();
           queue.getFunction("queueLenght", operand, r1.roadName, nvalue);
           cfunctionList.add(queue);
         return queue;
    }

   public  CFunction interuptLevel(String operand, Road r1, float value)
   {   CFunction interupt = new CFunction();
          interupt.getFunction("interuptLevel", operand, r1.roadName, value);
           cfunctionList.add(interupt);
         return interupt;
    }

      public  CFunction interuptLevel(String operand, Road r1, NValues nvalue)
   {   CFunction interupt = new CFunction();
           interupt.getFunction("interuptLevel", operand, r1.roadName, nvalue);
           cfunctionList.add(interupt);
         return interupt;
    }
//
//
//
   public CFunction getFunction(String functionTerm,
                                String operand, Road r1, float value, ArrayList flist) {
```

```
    CFunction funct = new CFunction();

      for(int index= 0;index<flist.size();index++ ){

        CFunction  f = (CFunction)flist.get(index);
      //if ((p.getName() == "intersect")
      && (p.getObj(0) == obj1)
      && (p.getObj(1) == obj1))
        //  return p;}
 if ((f.getTerminology().equals(functionTerm))
         && (f.getcOperator().equals(operand))
              && (f.getcName().equals(r1.roadName))
                     && (f.getcSigValue() == value))
         funct = f;
      }
              return funct;
}


  public CFunction getFunction(String functionTerm,
                               String operand, Road r1,
                               NValues nvalue, ArrayList flist)
                               {
    CFunction funct = new CFunction();

      for(int index= 0;index<flist.size();index++ ){

        CFunction  f = (CFunction)flist.get(index);
      //if ((p.getName() == "intersect")
      && (p.getObj(0) == obj1)
      && (p.getObj(1) == obj1))
        //  return p;}
 if ((f.getTerminology().equals(functionTerm))
```

```
            && (f.getcOperator().equals(operand))
                 && (f.getcName().equals(r1.roadName))
                        && (f.getcValue() == nvalue))
            funct = f;
        }
                 return funct;
  }
```

————Example of Java formated and Generated Domain Model

```
Domain Information
File Name: class basicplanner.domainFiles.DomainParameters
Domain Name:
Traffic Signal Control


Domain Predicates:
 (intersect  at_junction this_phase from_road1 to_road2)
 (JflowActive  at_junction this_phase from_road1 to_road2)
 (link  from_road1 to_road2)
 (RflowActive  from_road1 to_road2)


Domain Functions:
(=(queueLenght (this_road 10.0))
(=(capacity (this_road 200.0))
(=(interuptLevel (this_road 0.0))
```

Domain Operators:

```
(:process  Jtraffic_flow
:parameters [at_junction, this_phase, from_road1, to_road2]
:duration 5.0
:cost
:precondition [ (JflowActive  at_junction this_phase from_road1 to_road2)][][]
:effect ([][(decrease(queueLenght (from_road1 (* #10.0 1.0))),
            (increase(queueLenght (to_road2 (* #10.0 1.0)))][])
        not([][][])


(:process  Rtraffic_flow
:parameters [from_road1, to_road2]
:duration 5.0
:cost
:precondition [ (RflowActive  from_road1 to_road2)][][]
:effect ([][(decrease(queueLenght (from_road1 (* #10.0 1.0))),
            (increase(queueLenght (to_road2 (* #10.0 1.0)))][])
        not([][][])


(:action  switch_to_green
:parameters [at_junction, this_phase, from_road1, to_road2]
:duration 5.0
:cost
:precondition [ (intersect  at_junction this_phase from_road1 to_road2)]
            [(>(queueLenght (from_road1 0.0)),
            (<(interuptLevel (to_road2 7.0))][]
:effect ([ (JflowActive  at_junction this_phase from_road1 to_road2)][][])
        not([][][])
```

```
(:event  upstreamFilled
:parameters [at_junction, from_road1, to_road2]
:duration 1.0
:cost
:precondition [][(>=(queueLenght (to_road2 200.0))][]
:effect ([][(assign(queueLenght (to_road2 200.0)),
            (assign(interuptLevel (to_road2 7.0))][])
        not([][][])


(:event  road_linked
:parameters [from_road1, to_road2]
:duration 5.0
:cost
:precondition [ (link  from_road1 to_road2)]
                [(>(queueLenght (from_road1 0.0)),
                (<(interuptLevel (to_road2 7.0))][]
:effect ([ (RflowActive  from_road1 to_road2)][][])
        not([][][])



Total number of Predicate:4
Total number of numeric Functions:3
Total number of Operators:5


------------------------------------------------------------------------
BUILD SUCCESSFUL (total time: 0 seconds)
```

Appendix F: Sample UTCPLAN Problem File

Object file for the Road Traffic Network

```
Problem Information
File Name: class basicplanner.domainFiles.DomainParameters
Problem Name:  Traffic Problem

Domain Objects:

nLNorth   nLSouth   jWNorth   jWSouth   wDStr   lDSouth
brStr   lDNorth   pTStr   bmStr   pLNorth   pLSouth
Source   Sink   JunctionA   JunctionB   JunctionC
phase1   phase2

Domain Resources:


(:Resource  GreenPhase1
:parameters
lower_Bound: 1.0
upper_Bound: 20.0
weight: 0.05
)
```

```
(:Resource  GreenPhase2
:parameters
lower_Bound: 1.0
upper_Bound: 20.0
weight: 0.18
)

Domain Resource Constraints:


(:Constraints  GreenPhaseTime
:parameters
lower_Bound: 5.0
upper_Bound: 30.0
:linearFactors
GreenPhase1  10.0
GreenPhase2  5.0

)


(:Constraints  GreenPhaseTime
:parameters
lower_Bound: 5.0
upper_Bound: 30.0
:linearFactors
GreenPhase1  10.0
GreenPhase2  5.0
)


: Initial Condition
```

```
 (intersect  JunctionA phase1 nLNorth nLSouth)

 (intersect  JunctionA phase2 jWNorth jWSouth)

 (intersect  JunctionB phase1 wDStr lDSouth)

 (intersect  JunctionB phase2 brStr lDSouth)

 (intersect  JunctionC phase1 pTStr bmStr)

 (intersect  JunctionC phase2 pLNorth pLSouth)


 (link  nLSouth wDStr)

 (link  lDSouth pTStr)


(=(queueLenght (nLNorth 300.0))

(=(queueLenght (nLSouth 0.0))

(=(queueLenght (jWNorth 300.0))

(=(queueLenght (jWSouth 0.0))

(=(queueLenght (wDStr 0.0))

(=(queueLenght (lDSouth 0.0))

(=(queueLenght (brStr 300.0))
```

```
(=(queueLenght (lDNorth 0.0))

(=(queueLenght (pTStr 0.0))

(=(queueLenght (bmStr 0.0))

(=(queueLenght (pLNorth 0.0))

(=(queueLenght (pLSouth 0.0))

(=(interuptLevel (nLNorth 0.0))

(=(interuptLevel (nLSouth 0.0))

(=(interuptLevel (jWNorth 0.0))

(=(interuptLevel (jWSouth 0.0))

(=(interuptLevel (wDStr 0.0))

(=(interuptLevel (lDSouth 0.0))

(=(interuptLevel (brStr 0.0))

(=(interuptLevel (lDNorth 0.0))

(=(interuptLevel (pTStr 0.0))

(=(interuptLevel (bmStr 0.0))

(=(interuptLevel (pLNorth 0.0))

(=(interuptLevel (pLSouth 0.0))
```

```
: Goal Condition

(>(queueLenght (jWSouth 200.0))

(>(queueLenght (bmStr 200.0))



BUILD SUCCESSFUL (total time: 0 seconds)
```

Appendix G: Sample Thermal Problem and Solution Generated by the Planner

```
Domain Information

Domain Name:  ThermalControl

Domain Predicates:

 (on  ?heater)

 (in  ?water ?heater)

 (evap  ?water)

 (boil  ?water)

 (off  ?heater)

Domain Functions:

(=(temp 20.0))


Domain Operators:
```

```
(:action  switch_on

:parameters [heater]

:duration 1.0

:precondition [ (off  ?heater),  (in  ?water ?heater)]
                [(<(temp 60.0))][]

:effect ([ (on  ?heater)][(increase(temp 10.0))][])

not([ (off  ?heater)][][])


(:process  water

:parameters [water]

:duration 10.0

:precondition [ (on  ?heater),  (in  ?water ?heater)]
                    [(>(temp 60.0))][]

:effect ([][(increase(temp (* #10.0 1.0)))][])

not([][][])


(:event  boils

:parameters [water]

:duration 2.0
```

```
:precondition [ (on  ?heater),  (in  ?water ?heater)]
                    [(=(temp 100.0))][]

:effect ([(boil  ?water)][(increase(temp (* #10.0 1.0)))][])

not([][][])


(:process  boiling

:parameters [water]

:duration 10.0

:precondition [ (on  ?heater),  (in  ?water ?heater),
                (boil  ?water)][(>(temp 120.0))][]

:effect ([][(increase(temp 20.0))][])

not([][][])


(:event  evaporates

:parameters [water]

:duration 2.0

:precondition [ (in  ?water ?heater),  (boil  ?water),
                (on  ?heater)][(>(temp 130.0))][]

:effect ([(evap  ?water)][(increase(temp (* #10.0 1.0)))][])
```

```
not([ (boil  ?water),  (in  ?water ?heater)][][])


(:action  switch_off

:parameters [heater]

:duration 1.0

:precondition [ (on  ?heater),(evap  ?water)]
                [(>(temp 130.0))][]

:effect ([ (off  ?heater)][][])

not([ (on  ?heater)][][])


Initial Condition:

[ (off  ?heater),  (in  ?water ?heater)]

[]

Goal Condition:

[ (off  ?heater),  (evap  ?water)]

[]
```

Solution Plan Generated by the Planner

```
MAKESPAN OPERATOR STATUS PARAMETER DURATION
```

```
0.0 switch_on           [heater] 1.0

1.0     [process] water [water] 10.0

11.0 [event] boils [water] 2.0

13.0 [process] boiling [water] 10.0

23.0 [event] evaporates [water] 2.0

25.0 switch_off              [heater] 1.0


Timings in Nanoseconds, because it is too small for milliseconds

Total Searching Time is   86

Total Planning Time is  91

Preprocessing Time is   5

BUILD SUCCESSFUL (total time: ~0 seconds)
```

Appendix H: Sample Block World Problem and Solution Generated by the Planner

```
Domain Information

File Name: class basicplanner.domainFiles.DomainParameters

Domain Name: BlockWorld

Domain Predicates:

 (handEmpty  robot_hand)

 (on  x y)

 (clear  x)

 (holding  x)

 (onTable  x)

Domain Functions:
```

```
Domain Operators:

(:action  pick_up

:parameters [robot_hand, x]

:duration 1.0

:precondition [ (clear  x),  (onTable  x),
                (handEmpty  robot_hand)][][]

:effect ([ (holding  x)][][])

not([ (onTable  x),  (clear  x),
                (handEmpty  robot_hand)][][])


(:action  put_down

:parameters [robot_hand, x]

:duration 1.0

:precondition [ (holding  x)][][]

:effect ([ (onTable  x),  (clear  x),
                (handEmpty  robot_hand)][][])

not([ (holding  x)][][])

(:action  stack

:parameters [x, y, robot_hand]
```

```
:duration 1.0

:precondition [ (holding  x),(clear  y)][][]

:effect ([ (on  x y),  (clear  x),
                (handEmpty  robot_hand)][][])

not([ (clear  y),(holding  x)][][])

(:action  unstack

:parameters [x, y, robot_hand]

:duration 1.0

:precondition [ (on  x y),  (clear  x),
                (handEmpty  robot_hand)][][]

:effect ([ (holding  x),  (clear  y)][][])

not([ (on  x y),  (clear  x),
                (handEmpty  robot_hand)][][])


Total number of Predicate:5

Total number of numeric Functions:0

Total number of Operators:4

-------------------------------------------------------------------------------------------
```

```
_____

-------------------------------------------------------------------------------------------------------------

Problem Information

File Name: class basicplanner.domainFiles.DomainParameters

Problem Name:  fourBlock

Domain Objects:

A   B   C   D   robot_Hand


: Initial Condition

 (clear  A)

 (clear  B)

 (clear  C)

 (clear  D)

 (onTable  A)

 (onTable  B)

 (onTable  C)

 (onTable  D)

 (handEmpty  robot_hand)
```

```
: Goal Condition

 (on  A B)

 (on  C D)
```

Solution Plan Generated by the Planner

```
MAKESPAN OPERATOR STATUS PARAMETER DURATION

0.0         pick_up [robot_hand, A] 1.0

1.0         stack [A, B, robot_hand] 1.0

2.0         pick_up [robot_hand, C] 1.0

3.0         stack [C, D, robot_hand] 1.0
```

References

[1] W. Al-Gherwi, H. Budman, and A. Elkamel. A robust distributed model predictive control algorithm. *Journal of Process Control*, 21(8):1127–1137, 2011. 68

[2] R. Alterovitz, T. Simeon, and K. Goldberg. The stochastic motion roadmap: A sampling framework for planning with markov motion uncertainty. In *Proceedings of Robotics: Science and Systems*, Atlanta, GA, USA, June 2007. 20

[3] A. Armando, E. Giunchiglia, M. Maratea, and S. E. Ponta. An action-based approach to the formal specification and automatic analysis of business processes under authorization constraints. *Journal of Computer and System Sciences*, 78(1):119 – 141, 2012. {JCSS} Knowledge Representation and Reasoning. 38

[4] G. Audemard, P. Bertoli, A. Cimatti, A. Kornilowicz, and R. Sebastiani. A SAT-based approach for solving formulas over boolean and linear mathematical propositions. In *Proc. 18th Int. Conf. on Automated Deduction*, volume 2392, pages 193–208. Springer-Verlag, LNAI Series, 2002. 27, 116

[5] K. Awahara, S. Izumi, T. Abe, and T. Suganuma. Autonomous control method using ai planning for energy-efficient network systems. In *Broadband and Wireless Computing, Communication and Applications (BWCCA), 2013 Eighth International Conference on*, pages 628–633, Oct 2013. 38

[6] J. Backman, T. Oksanen, and A. Visala. Navigation system for agricultural machines: Nonlinear model predictive path tracking. *Computers and Electronics in Agriculture*, 82:32–43, 2012. 68

[7] P. G. Balaji and D. Srinivasan. Multi-agent system in urban traffic signal control. *IEEE Computational Intelligence Magazine*, 5(4):43–51, 2010. 49

[8] J. M. Barnes, A. Pandey, and D. Garlan. Automated planning for software architecture evolution. pages 213–223. IEEE, 2013. 36

[9] L. Batrinca, M. T. Khan, D. Billman, B. Aydemir, and G. Convertino. A timeline visualization for multi-team collaborative planning. In *CHI '13 Extended Abstracts on Human Factors in Computing Systems*, CHI EA '13, pages 157–162, New York, NY, USA, 2013. ACM. 24

[10] A. L. Bazzan. A distributed approach for coordination of traffic signal agents. *Autonomous Agents and Multi-Agent Systems*, 10(1):131–164, Jan. 2005. 98

[11] S. Bennett. *A History of Control Engineering 1930-1955*. Peter Peregrinus, Hitchin, Herts., UK, UK, 1st edition, 1993. 68

[12] J. Benton, A. J. Coles, and A. Coles. Temporal planning with preferences and time-dependent continuous costs. In *ICAPS*, 2012. 63

[13] P. Bercher, S. Biundo, T. Geier, T. Hoernle, F. Nothdurft, F. Richter, and B. Schattenberg. Plan, repair, execute, explain - how planning helps to assemble your home theater. In *Proceedings of the Twenty-Fourth International Conference on Automated Planning and Scheduling, ICAPS 2014, Portsmouth, New Hampshire, USA, June 21-26, 2014*, 2014. 40

[14] J.-C. Bermond, N. Marlin, D. Peleg, and S. Pérennes. Directed virtual path layout in ATM networks. *Theoretical Computer Science*, 291(1):3–28, 2003. 37

[15] S. Bernardini and K. Porayska-Pomsta. Planning-based social partners for children with autism, 2013. 37

[16] D. L. Berre, P. Marquis, and S. Roussel. Planning personalised museum visits, 2013. 39

[17] A. Blum and M. Furst. Fast planning through planning graph analysis. In *Proceedings of the International Joint Conference on Artificial Inteligence (IJCAI-95)*, 1995. 28

[18] D. Borrajo, S. Kambhampati, A. Oddi, and S. Fratini, editors. *Proceedings of the Twenty-Third International Conference on Automated Planning and Scheduling, ICAPS 2013, Rome, Italy, June 10-14, 2013*. AAAI, 2013. 42

[19] J. L. Bresina, R. Dearden, N. Meuleau, S. Ramakrishnan, D. E. Smith, and R. Washington. Planning under continuous time and resource uncertainty: A challenge for ai. *CoRR*, abs/1301.0559, 2013. 27

[20] E. Camacho and C. Bordons. *Model predictive control*. Springer, London, 1999. 68, 87, 96

[21] E. Camponogara and L. B. de Oliveira. Distributed optimization for model predictive control of linear-dynamic networks. *IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans*, 39(6):1331–1338, 2009. 49

[22] R. Castano, T. Estlin, D. Gaines, C. Chouinard, B. Bomstein, R. Anderson, M. Burl, D. Thompson, A. Castano, and M. Judd. Onboard autonomous rover science. In *Aerospace Conference, 2007 IEEE*, pages 1 –13, march 2007. 36

[23] L. Castillo, E. Armengol, E. Onaindía, L. Sebastiá, J. González-Boticario, A. Rodríguez, S. Fernández, J. D. Arias, and D. Borrajo. samap: An user-oriented adaptive system for planning tourist visits. *Expert Syst. Appl.*, 34(2):1318–1332, Feb. 2008. 39

[24] L. A. Castillo, L. Morales, A. González-Ferrer, J. Fernández-Olivares, D. Borrajo, and E. Onaindia. Automatic generation of temporal planning domains for e-learning problems. *J. Scheduling*, pages 347–362, 2010. 39

[25] A. Cesta, G. Cortellessa, S. Fratini, and A. Oddi. Developing an end-to-end planning application from a timeline representation framework. In *IAAI*, 2009. 24

[26] S. Chien, B. Smith, G. Rabideau, N. Muscettola, and K. Rajan. Automated planning and scheduling for goal-based autonomous spacecraft. *IEEE Intelligent Systems*, 13:50–55, 1998. 36

[27] A. H. W. Chun. Optimizing limousine service with ai. *AI Magazine*, 32(2):27–41, 2011. 38

[28] A. Cimatti, F. Giunchiglia, E. Giunchiglia, and P. Traverso. Planning via model checking: A decision procedure for *r*. In S. Steel and R. Alami, editors, *ECP*, volume 1348 of *Lecture Notes in Computer Science*, pages 130–142. Springer, 1997. 28

[29] M. Cirillo, T. Uras, and S. Koenig. A lattice-based approach to multi-robot motion planning for non-holonomic vehicles. pages 232–239. IEEE, 2014. 41

[30] A. I. Coles, M. Fox, K. Halsey, D. Long, and A. J. Smith. Managing concurrency in temporal planning using planner-scheduler interaction. *Art. Int. (AIJ)*, 173:1–44, 2008. 25, 26

[31] A. I. Coles, M. Fox, D. Long, and A. J. Smith. A Hybrid Relaxed Planning Graph-LP Heuristic for Numeric Planning Domains. In *Proc. 18th Int. Conf. on Automated Planning and Scheduling (ICAPS)*, 2008. 24

[32] A. I. Coles, M. Fox, D. Long, and A. J. Smith. Planning with problems requiring

temporal coordination. In *Proc. 23rd AAAI Conf. on Artificial Intelligence*, July 2008. 26, 29, 63

[33] A. J. Coles, A. Coles, M. Fox, and D. Long. Colin: Planning with continuous linear numeric change. *J. Artif. Intell. Res. (JAIR)*, 44:1–96, 2012. 29

[34] A. J. Coles and A. I. Coles. Lprpg-p: Relaxed plan heuristics for planning with preferences. In *Proceedings of the Twenty First International Conference on Automated Planning and Scheduling (ICAPS-11)*, June 2011. 19

[35] A. J. Coles, A. I. Coles, M. Fox, and D. Long. Planning with linear continuous numeric change. In *Proceedings of the 27th Workshop of the UK Planning and Scheduling Special Interest Group (PlanSIG 2008)*, December 2008. 2

[36] A. J. Coles, A. I. Coles, M. Fox, and D. Long. Forward-chaining partial-order planning. In *Proceedings of the Twentieth International Conference on Automated Planning and Scheduling (ICAPS-10)*, May 2010. 19

[37] G. Cortellessa, R. D. Benedictis, and M. Pagani. Timeline-based planning for engaging training experiences, 2013. 39

[38] R. V. Cowlagi and P. Tsiotras. Hierarchical motion planning with kinodynamic feasibility guarantees: Local trajectory planning via model predictive control. pages 4003–4008. IEEE, 2012. 68

[39] P. A. De Clercq, J. A. Blom, H. H. M. Korsten, and A. Hasman. Approaches for creating computer-interpretable guidelines that facilitate decision support. *Artificial Intelligence in Medicine*, 31(1):1–27, 2004. 37

[40] D. De Oliveira and A. L. C. Bazzan. *Multiagent Learning on Traffic Lights Control: Effects of Using Shared Information*, pages 307–322. 2009. 52

[41] C. Dimon and A. I. Udrea. Compartmental networks approach on urban traffic control. pages 166–171. IEEE, 2013. 49

[42] M. B. Do and S. Kambhampati. Sapa: A Multi-objective Metric Temporal Planner. *J. Art. Int. Res. (JAIR)*, 20:155–194, 2003. 2

[43] M. B. Do and S. Kambhampati. Sapa: A scalable multi-objective heuristic metric temporal planner. *Journal of Artificial Intelligence Research*, 20:2003, 2003. 25

[44] C. Domshlak, J. Hoffmann, and A. Sabharwal. Friends or Foes? On Planning as Satisfiability and Abstract CNF Encodings. *J. Art. Int. Res. (JAIR)*, 36:415–469, 2009. 20

[45] B. Durmus, H. Temurtas, N. Yumusak, and F. Temurtas. A study on industrial robotic manipulator model using model based predictive controls. *Journal of Intelligent Manufacturing*, 20(2):233–241, 2009. 68

[46] I. Dusparic and V. Cahill. Autonomic multi-policy optimization in pervasive systems: Overview and evaluation. *ACM Trans. Auton. Adapt. Syst.*, 7(1), May 2012. 98

[47] S. Edelkamp and S. Jabbar. Cost-optimal external planning. In *Proceedings of In 21st National (American) Conference on Artificial Intelligence (AAAI)*. AAAI Press, July 2006. 25

[48] P. Eyerich, R. Mattmüller, and G. Röger. Using the Context-enhanced Additive Heuristic for Temporal and Numeric Planning. In *Proc. 19th Int. Conf. on Automated Planning and Scheduling (ICAPS)*, 2009. 26

[49] P. Falcone, F. Borrelli, H. E. Tseng, J. Asgari, and D. Hrovat. A hierarchical model predictive control framework for autonomous ground vehicles. pages 3719–3724. IEEE, 2008. 68

[50] P. Falugi, S. Olaru, and D. Dumur. Robust multi-model predictive control using lmis. *International Journal of Control, Automation and Systems*, 8(1):169–175, 2010. 68

[51] J. Fdez-Olivares, L. Castillo, J. A. Cózar, and O. García Pérez. Supporting clinical processes and decisions by hierarchical planning and scheduling. *Computational Intelligence*, 27(1):103–122, 2011. 37

[52] A. G. Ferrer. *Knowledge Engineering Techniques for the Translation of Process Models into Temporal Hierarchical Planning and Scheduling Domains*. Phd thesis, 2011. 38

[53] R. E. Fikes and N. J. Nilsson. Strips: A new approach to the application of theorem proving to problem solving. In *Proceedings of the 2Nd International Joint Conference on Artificial Intelligence*, IJCAI'71, pages 608–620, San Francisco, CA, USA, 1971. Morgan Kaufmann Publishers Inc. 22, 30

[54] J. E. Flórez, Á. T. A. de Reyna, J. García, C. Linares López, A. G. Olaya, and D. Borrajo. Planning multi-modal transportation problems. In *Proceedings of the 21st International Conference on Automated Planning and Scheduling, ICAPS 2011, Freiburg, Germany June 11-16, 2011*, 2011. 103

[55] I. Foster and C. Kesselman, editors. *The Grid: Blueprint for a New Computing Infrastructure*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1999. 44

[56] M. Fox, R. Howey, and D. Long. Validating plans in the context of processes and exogenous events. In M. M. Veloso and S. Kambhampati, editors, *AAAI*, pages 1151–1156. AAAI Press / The MIT Press, 2005. 28

[57] M. Fox and D. Long. PDDL2.1: An extension of PDDL for expressing temporal planning domains. *J. Art. Int. Res. (JAIR)*, 20:61–124, 2003. 11, 32, 60

[58] M. Fox and D. Long. Modelling mixed discrete-continuous domains for planning. *J. Art. Int. Res. (JAIR)*, 27:235–297, 2006. 27, 29, 104

[59] M. Fox, D. Long, and D. Magazzeni. Plan-based policy-learning for autonomous feature tracking. 2012. 95

[60] M. Fox, D. Long, and D. Magazzeni. Plan-based policies for efficient multiple battery load management. *CoRR*, abs/1401.5859, 2014. 98

[61] J. Frank. *Planning solar array operations on the international space station*, pages 470–471. 2013. 37

[62] I. Georgievski, T. A. Nguyen, and M. Aiello. Combining activity recognition and ai planning for energy-saving offices. In *Ubiquitous Intelligence and Computing, 2013 IEEE 10th International Conference on and 10th International Conference on Autonomic and Trusted Computing (UIC/ATC)*, pages 238–245, Dec 2013. 38

[63] A. Gerevini, P. Haslum, D. Long, A. Saetti, and Y. Dimopoulos. Deterministic planning in the fifth international planning competition: PDDL3 and experimental evaluation of the planners. *Art. Int. (AIJ)*, 173(5-6):619–668, 2009. 11

[64] A. Gerevini, A. Saetti, and I. Serina. An approach to temporal planning and scheduling in domains with predictable exogenous events. *J. of AI Research*, 25:187–231, 2006. 24, 63

[65] M. Gopal. *Control systems: principles and design*. McGraw-Hill, London, 2008. 87

[66] D. Grether, A. Neumann, and K. Nagel. Simulation of urban traffic control: A queue model approach. *Procedia Computer Science*, 10:808–814, 2012. 49

[67] C. Guo, X. Gang, and M. Zhang. Model predictive control implementation and simulation for urban traffic networks. In *Service Operations and Logistics, and Informatics (SOLI), 2014 IEEE International Conference on*, pages 334–340, Oct 2014. 76

[68] S. K. Gupta, D. S. Nau, and W. C. Regli. IMACS: A case study in real-world planning. *IEEE Expert and Intelligent Systems*, 13(3):49–60, 1998. 21

[69] A. Hamilton, B. Waterson, T. Cherrett, A. Robinson, and I. Snell. The evolution of urban traffic control: changing policy and technology. *Transportation Planning and Technology*, 36(1):24–43, 2013; 2012. 43, 44, 47

[70] P. Haslum and H. Geffner. Heuristic planning with time and resources. In *Proc. 6th European Conference on Planning (ECP'01)*, pages 121–132, 2001. 25

[71] O. Hatzi, D. Vrakas, N. Bassiliades, D. Anagnostopoulos, and I. Vlahavas. The porsce ii framework: Using ai planning for automated semantic web service composition. *Knowledge Engineering Review*, 28(2):137–156, 2013. 39

[72] M. Helmert. The fast downward planning system. *J. Art. Int. Res. (JAIR)*, 26:191–246, 2006. 26

[73] J. Hoffmann. The Metric-FF Planning System: Translating "Ignoring Delete Lists" to Numeric State Variables. *J. Art. Int. Res. (JAIR)*, 20:291–341, 2003. 2, 24

[74] J. Hoffmann and B. Nebel. The FF planning system: Fast plan generation through heuristic search. *J. Art. Int. Res. (JAIR)*, 14:253–302, 2001. 19, 23, 26

[75] S. P. Hoogendoorn and P. H. L. Bovy. State-of-the-art of vehicular traffic flow modelling. In *Delft University of Technology, Delft, The*, pages 283–303, 2001. 76

[76] R. Huang, Y. Chen, and W. Zhang. An Optimal Temporally Expressive Planner: Initial Results and Application to P2P Network Optimization. In *International Conference on Automated Planning and Scheduling*, 2009. 26

[77] S. Ibarra-Martnez, J. A. Castn-Rocha, and J. Laria-Menchaca. Optimizing urban traffic control using a rational agent. *Journal of Zhejiang University SCIENCE C*, 15(12):1123–1137, 2014. 49

[78] S. Jimnez, F. Fernndez, and D. Borrajo. Integrating planning, execution, and learning to improve plan execution. *Computational Intelligence*, 29(1):1–36, 2013. 86

[79] F. Jimoh, L. Chrpa, T. McCluskey, and M. M. S. Shah. Towards application of automated planning in urban traffic control. In *16th International IEEE Conference on Intelligent Transportation Systems*, October 2013. automated planning, urban transport control, autonomic systems. 52, 57

[80] H. A. Kautz. Deconstructing planning as satisfiability. In *Proc. 21st Nat. Conf. on Artificial Intelligence (AAAI)*, 2006. 20

[81] H. A. Kautz and B. Selman. Pushing the Envelope: Planning, Propositional Logic and Stochastic Search. In *Proc. of AAAI Int. Conf. on AI*, pages 1194–1201, 1996. 20

[82] O. Kaynak, P. Melancon, and V. Rajagopalan. Model predictive heuristic control of a position servo system in robotics application. volume 3, pages 481–485. IEEE, 1987. 68

[83] J. Koehler and D. Ottiger. An ai-based approach to destination control in elevators. *AI Magazine*, 23(3):59–59, 2002. 40

[84] A. Kotsialos, I. Papamichail, I. Margonis, and M. Papageorgiou. Hierarchical nonlinear model-predictive ramp metering control for freeway networks. In *Proceedings of the 11th IFAC Symposium on Control in Transportation Systems, Delft, The Netherlands*, pages 124–129, 2006. 48

[85] A. Kovcs. Task sequencing for remote laser welding in the automotive industry, 2013. 41

[86] N. Kushmerick, S. Hanks, and D. S. Weld. An algorithm for probabilistic planning. *Artificial Intelligence*, 76(12):239 – 286, 1995. Planning and Scheduling. 21

[87] U. Kuter and D. Nau. Forward-chaining planning in nondeterministic domains, 2004. 19

[88] J. O. Laguna, A. G. Olaya, and D. B. Millan. *Building Planning Action Models Using Activity Recognition*. PhD thesis, Universidad Carlos III de Madrid, 2014. 85, 93

[89] T. Le, H. L. Vu, Y. Nazarathy, B. Vo, and S. Hoogendoorn. Linear-quadratic model predictive control for urban traffic networks. *Procedia - Social and Behavioral Sciences*, 80:512–530, 2013. 47, 49

[90] T. Le, H. L. Vu, Y. Nazarathy, B. Vo, and S. Hoogendoorn. Linear-quadratic model predictive control for urban traffic networks. *Procedia - Social and Behavioral Sciences*, 80(0):512 – 530, 2013. 20th International Symposium on Transportation and Traffic Theory (ISTTT 2013). 68

[91] S. Lemai and F. Ingrand. Interleaving temporal planning and execution: Ixtet-exec, 2003. 24

[92] C. Leung, S. Huang, N. Kwok, and G. Dissanayake. Planning under uncertainty using model predictive control for information gathering. *Robotics and Autonomous Systems*, 54(11):898–910, 2006. 68

[93] B. Li and M. Riedl. Creating customized game experiences by leveraging human creative effort: A planning approach. In F. Dignum, editor, *Agents for Games*

and Simulations II, volume 6525 of *Lecture Notes in Computer Science*, pages 99–116. Springer Berlin / Heidelberg, 2011. 10.1007/978-3-642-18181-8_8. 40

[94] H. Li and B. Williams. Generative systems for hybrid planning based on flow tubes. In *Proc. 18th Int. Conf. on Aut. Planning and Scheduling (ICAPS)*, 2008. 28

[95] H. Li and B. Williams. Hybrid planning with temporally extended goals for sustainable ocean observing. In *Proceedings of the Twenty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2011, San Francisco, California, USA, August 7-11, 2011*, 2011. 28

[96] D. Long and M. Fox. The 3rd International Planning Competition: Results and Analysis. *Journal of Artificial Intelligence Research*, 20:1–59, 2003. 131

[97] T. L. McCluskey. Object Transition Sequences: A New Form of Abstraction for HTN Planners. 2000. 31

[98] T. L. McCluskey, D. Liu, and R. M. Simpson. GIPO II: HTN Planning in a Tool-supported Knowledge Engineering Environment. In *Proceedings of the Thirteenth International Conference on Automated Planning and Scheduling*, pages 92 – 101. AAAI Press, Menlo Park, California, 2003. 31

[99] D. McDermott. Reasoning about Autonomous Processes in an Estimated Regression Planner. In *Proc. 13th Int. Conf. on Aut. Planning and Scheduling (ICAPS)*, 2003. 27

[100] McDermott D. et al. PDDL–the planning domain definition language. Technical report, Available at: `www.cs.yale.edu/homes/dvm`, 1998. 31, 90

[101] C. McGann, F. Py, K. Rajan, H. Thomas, R. Henthorn, and R. McEwen. A Deliberative Architecture for AUV Control. In *Intnl. Conf. on Robotics and Automation (ICRA)*, Pasadena, May 2008. 85

[102] N. Meuleau, E. Benazera, R. I. Brafman, E. A. Hansen, and Mausam. A heuristic search approach to planning with continuous resources in stochastic domains. *J. Artif. Intell. Res. (JAIR)*, 34:27–59, 2009. 29

[103] E. Mitsakis, J. M. Salanova, and G. Giannopoulos. Combined dynamic traffic assignment and urban traffic control models. *Procedia - Social and Behavioral Sciences*, 20:427–436, 2011. 47

[104] C. Muise, S. A. McIlraith, and J. C. Beck. Monitoring the Execution of Partial-Order Plans via Regression. In *International Joint Conference On Artificial Intelligence*, International Joint Conference On Artificial Intelligence, 2011. 85

[105] K. L. Myers. Towards a framework for continuous planning and execution. In *In Proceedings of the AAAI Fall Symposium on Distributed Continual Planning*. AAAI Press, 1998. 85

[106] N. S. Nafi, R. H. Khan, J. Y. Khan, and M. Gregory. A predictive road traffic management system based on vehicular ad-hoc network. pages 135–140. IEEE, 2014. 48

[107] A. Nareyek, E. C. Freuder, R. Fourer, E. Giunchiglia, R. P. Goldman, H. A. Kautz, J. Rintanen, and A. Tate. Constraints and ai planning. *IEEE Intelligent Systems*, pages 62–72, 2005. 40

[108] D. Nau, M. Ghallab, and P. Traverso. *Automated Planning: Theory & Practice*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2004. 95

[109] F. Palao, L. Castillo, J. Fdez-Olivares, and O. García. Cities that offer a customized and personalized tourist experience to each and every visitor: the smartourism project. In *Proceedings of the AI for an Intelligent Planet*, AIIP '11, pages 4:1–4:8, New York, NY, USA, 2011. ACM. 39

[110] M. Papageorgiou, C. Diakaki, V. Dinopoulou, A. Kotsialos, and Y. Wang. Review of road traffic control strategies. *Proceedings of the IEEE*, 91(12):2043–2067, Dec 2003. 49

[111] S. Parkinson, A. Longstaff, A. Crampton, and P. Gregory. The application of automated

planning to machine tool calibration. In L. McCluskey, B. Williams, J. R. Silva, and B. Bonet, editors, *ICAPS*. AAAI, 2012. 41

[112] S. Parkinson, A. P. Longstaff, and S. Fletcher. Automated planning to minimise uncertainty of machine tool calibration. 2014. 41

[113] S. Penberthy and D. Weld. Temporal Planning with Continuous Change. In *Proc. 12th Nat. Conf. on AI (AAAI)*, pages 1010–1015. AAAI/MIT Press, 1994. 27

[114] G. D. Penna, B. Intrigila, D. Magazzeni, and F. Mercorio. Upmurphi: a tool for universal planning on pddl+ problems. In *Proc. 19th Int. Conf. on Automated Planning and Scheduling (ICAPS)*, pages 19–23, 2009. 131

[115] G. D. Penna, B. Intrigila, D. Magazzeni, and F. Mercorio. A pddl+ benchmark problem: The batch chemical plant. In *ICAPS'10*, pages 222–225, 2010. 28

[116] G. D. Penna, D. Magazzeni, F. Mercorio, and B. Intrigila. Upmurphi: A tool for universal planning on pddl+ problems. In A. Gerevini, A. E. Howe, A. Cesta, and I. Refanidis, editors, *ICAPS*. AAAI, 2009. 28, 36

[117] E. Perea-López, B. E. Ydstie, and I. E. Grossmann. A model predictive control strategy for supply chain optimization. *Computers and Chemical Engineering*, 27(8):1201–1218, 2003. 68

[118] R. P. A. Petrick and M. E. Foster. Planning for social interaction in a robot bartender domain. In *Proceedings of the International Conference on Automated Planning and Scheduling (ICAPS 2013), Special Track on Novel Applications*, Rome, Italy, June 2013. 41

[119] C. Piacentini, V. Alimisis, M. Fox, and D. Long. Combining a temporal planner with an external solver for the power balancing problem in an electricity network, 2013. 41

[120] J. Pinto, J. Sousa, F. Py, and K. Rajan. Experiments with Deliberative Planning on Autonomous Underwater Vehicles. In *IROS Workshop on Robotics for Environmental Modeling*, Algarve, Portugal, 2012. 85

[121] N. Policella, H. Oliveira, and E. Benzi. Planning spacecraft activities: An automated approach, 2013. 37

[122] S. Richter and M. Westphal. The LAMA Planner: Guiding Cost-Based Anytime Planning with Landmarks. *Journal of Artificial Intelligence Research*, To Appear, 2010. 15

[123] D. A. Roozemond. Using intelligent agents for pro-active, real-time urban intersection control. *European Journal of Operational Research*, 131(2):293–301, 2001. 52

[124] W. Ruml, M. B. Do, and M. P. J. Fromherz. Online planning and scheduling for high-speed manufacturing. In *In Proc. ICAPS Workshop on Integrating Planning into Scheduling*, pages 2–11. AAAI Press, 2005. 41

[125] S. Sanner. Relational dynamic influence diagram language (RDDL): Language description. http://users.cecs.anu.edu.au/ ssanner/IPPC_2011/RDDL.pdf, 2010. 31

[126] J.-A. Shin and E. Davis. Processes and continuous change in a sat-based planner. *Artif. Intell.*, 166(1-2):194–253, 2005. 20, 116

[127] G. Slager, G. Slager, M. Milano, and M. Milano. Urban traffic control system using self-organization. pages 255–260. IEEE, 2010. 86

[128] S. Smaili, D. Ichalal, F. Chu, S. Mammar, and S. Mammar. Model predictive control for an urban trimodal model. pages 356–361. IEEE, 2012. 48

[129] B. D. Smith, K. Rajan, and N. Muscettola. Knowledge acquisition for the onboard planner of an autonomous spacecraft. In *Knowledge Acquisition, Modeling and Management, 10th European Workshop, EKAW'97*, pages 253–268, 1997. 15, 24

[130] D. Smith and D. S. Weld. Temporal Planning with Mutual Exclusion Reasoning. In *Proc. 16th Int. Joint Conf. on AI (IJCAI)*, pages 326–337, 1999. 25

[131] D. E. Smith and D. S. Weld. Temporal planning with mutual exclusion reasoning. In T. Dean, editor, *IJCAI*, pages 326–337. Morgan Kaufmann, 1999. 20

[132] S. Smith, G. Barlow, X.-F. Xie, and Z. Rubinstein. Surtrac: Scalable urban traffic control. In *Transportation Research Board 92nd Annual Meeting Compendium of Papers*. Transportation Research Board, January 2013. 49

[133] S. Sohrabi, O. Udrea, and A. Riabov. Htn planning for the composition of stream processing applications. In *Proceedings of the 23rd International Conference on Automated Planning and Scheduling - Novel Applications Track (ICAPS-13)*, pages 443–451, Rome, Italy, June 2013. 36

[134] B. Srivastava, J. P. Bigus, and D. A. Schlosnagle. Bringing planning to autonomic applications with able. In *Proceedings of the First International Conference on Autonomic Computing*, ICAC '04, pages 154–161, Washington, DC, USA, 2004. IEEE Computer Society. 38

[135] M. Tay. Model predictive cost control. *Control Engineering*, 54(8):IE9, 2007. 68

[136] S. Tzafestas, G. Kapsiotis, and E. Kyriannakis. Model-based predictive control for generalized production planning problems. *Computers in Industry*, 34(2):201–210, 1997. 68

[137] M. Vallati. A guide to portfolio-based planning. In C. Sombattheera, N. K. Loi, R. Wankar, and T. T. Quan, editors, *MIWAI*, volume 7694, pages 57–68. Springer, 2012. 21

[138] J. Van Den Berg, S. Patil, and R. Alterovitz. Motion planning under uncertainty using iterative local optimization in belief space. *Int. J. Rob. Res.*, 31(11):1263–1278, Sept. 2012. 21

[139] I. Varga, B. Kulcsar, T. Luspay, T. Peni, T. Tettamanti, C. U. of Technology, C. tekniska hgskola, R. Institutionen fr signaler och system, D. of Signals, and A. C. Systems. Robust real-time control for urban road traffic networks. *IEEE transactions on intelligent transportation systems*, 15(1):385, 2014. 47

[140] V. Veselý, D. Rosinov, and M. Foltin. Robust model predictive control design with input constraints. *ISA Transactions*, 49(1):114–120, 2010. 68, 69

[141] V. Vidal and H. Geffner. Branching and pruning: An optimal temporal pocl planner based on constraint programming. *Artif. Intell.*, 170(3):298–335, 2006. 25

[142] Y. Wang, D. Wang, B. Xu, and T. Wongpiromsarn. Junction-based model predictive control for urban traffic light control. pages 54–59. IEEE, 2013. 47

[143] M. Wehrle and J. Rintanen. Planning as Satisfiability with Relaxed ∃-Step Plans. In *Proc. Australian Conf. on Art. Int.*, pages 244–253, 2007. 20

[144] A. Winder, M. Brackstone, P. D. Site, and M. Antognoli. Research to increase the capacity, efficiency, sustainability and safety of road, rail and urban transport networks. Technical report, Transport Research Knowledge Center, Traffic Management for Land Transport, June 2009. 44

[145] S. Wolfman and D. Weld. The LPSAT System and its Application to Resource Planning. In *Proc. 16th Int. Joint Conf. on Artificial Intelligence (IJCAI)*, 1999. 27, 36

[146] J. Yang, L. Zhang, Y. Chen, and L. Shi. Modeling and control of signaling split in urban traffic network based on hybrid systems. volume 7, pages 3497–3502, 2010. 48

[147] S. Yoon. Learning heuristic functions from relaxed plans. In *In ICAPS*. AAAI Press, 2006. 19

[148] H. L. S. Younes, M. L. Littman, D. Weissman, and J. Asmuth. The First Probabilistic Track of the International Planning Competition. *J. Art. Int. Res. (JAIR)*, 24:851–887, 2005. 31

[149] H. L. S. Younes and R. G. Simmons. VHPOP: Versatile heuristic partial order planner. *Journal of Artificial Intelligence Research (JAIR)*, 20:405–430, 2003. 25