Enhanced Heuristics for Numeric Temporal Planning

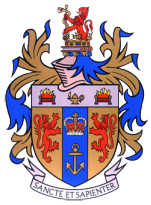
Piacentini, Chiara

Awarding institution:
King's College London

King's College London

Faculty of Natural & Mathematical Science

Department of Informatics



Enhanced Heuristics for Numeric Temporal Planning

Chiara Piacentini

A thesis submitted in partial fulfilment of the requirements of the degree of Doctor of Philosophy

First Supervisor: Prof. Maria FOX
Second Supervisor: Prof. Derek LONG

July 2015

Abstract

After 50 years of fundamental research, domain independent planning has recently started to be applied to numerous real world problems. However, this has shown that the techniques developed until now are not completely mature: improvements can be made in different directions, such as in the area of metric temporal planning. This PhD research is focused on how we can use more sophisticated and informative heuristics in the general context of automated planning, when numeric and temporal constraints are a significant part of the problem.

As a starting point, we will use as a reference example the voltage control problem in distributed electricity networks of power systems. This domain is a real world application of planning in which non-linear numeric effects and exogenous events are combined, challenging the *state of the art* planners. A power system is a nation-wide infrastructure which delivers electricity from suppliers to consumers. Technical and economic considerations impose constraints on the different elements of the network and subsequently on control and controlling variables of interest. One of the main parameters is the voltage, which must lie in strict boundaries. The voltage, as well as all the other physical quantities involved, is subject to the variation of the electrical output that changes through time. Effects of these changes propagate all across the network. This comes with a substantial computational burden and calls for extensions to be developed to enable the application of automated planning. In addition to this, the presence of events representing predicted load and supply over time require the planner to interact with uncontrollable events.

The voltage control problem is the starting point for our investigation on how the standard delete-relaxation behaves in the presence of numeric and temporal constraints. Fully relaxing all the negative effects can result in a too poorly informed heuristic. In this thesis we explore different ways to enhance the heuristic, adding selected negative effects, while not compromising too much the efficiency of the heuristic computation. In particular we have studied the numeric temporal heuristic of the planner POPF2, based on the Temporal Relaxed Planning Graph (TRPG), and propose a way to take into account numeric effects that are calculated by external modules connected to the planner. Negative effects of predictable numeric exogenous events in the presence of trajectory constraints are also taken into account in the heuristic.

Acknowledgement

This work would have not been the same without the contribution of many people.

First and foremost, I would like to express my deepest gratitude to my supervisors Maria Fox and Derek Long. Their invaluable guidance, extraordinary support, and constant advice were fundamental throughout these years.

I would like also to thank my examiners Barry O'Sullivan and Franco Raimondi for the useful comments and interesting discussion.

My Ph.D. studies were supported by the EPSRC, which I would like to thank. The Autonomic Power System project was very inspiring and my gratitude goes towards all the members of the project for the exchange of ideas and visions. Particularly, I would like to mention Barbara: her patience and insights were essential to my understanding of the electrical domain.

I greatly appreciate the feedback received during these years by the AI planning community, particularly by my ICAPS mentor Sylvie Thiébaux and all the members, past and present, of the King's College Planning Group.

Last but not least, I would like to express a special thank you to all the unnamed people who were close to me in these years.

Contents

List of Figures

List of Tables

List of Algorithms

Chapter 1

Introduction

In recent years, an important part of the research in Automated Planning has focused on "real-world" applications, as evidenced by the introduction of the special *application track* in the ICAPS Conference (International Conference of Automated Planning and Scheduling). On one hand, this shows that planning has reached enough maturity to be applied to "real-world" problems, on the other hand, great efforts are made by researchers to integrate standard tools or develop new ones into their applications. One of the most recent examples is the work in (Cashmore et al., 2014), where planning is used by autonomous underwater vehicles (AUVs) for the autonomous inspection and maintenance of a seabed facility. In this work a domain independent planner (POPF2) is used to generate high level temporal plans, which are integrated in an execution framework. If the execution of an action fails, then a new problem model is generated, producing a new plan. Another example of a planning application can be found in (Bercher et al., 2014). In this work, planning is combined with other components that provide generation, execution, repair, and explanation of plans for user assistance in assembly of technical systems, such as home theatres. Planning is also used in autonomous planetary exploration missions, such as in the ESA project GOAC (Ceballos et al., 2011). The system described in this paper is composed of two different layers: the lower layer (*functional layer*) deals with evolving conditions that the system has to react to, while the higher layer deals with long-term mission plans, which are generated and dispatched in an intertwined way, with the T-REX system (Py et al., 2010).

The original problem that motivated this research is an application of planning techniques to the pro-active management of operations in distribution networks (Piacentini et al., 2013). The problem consists of ensuring that the electricity demand of the consumers is met by the committed supply, while numeric and temporal constraints are imposed on the different elements of the network so that damage to the equipment is prevented. This domain falls in a new class of metric temporal planning problems, characterised by the need to manage both plan trajectory constraints and uncontrollable numeric events. In addition, this domain presents important modelling challenges, because the voltage and other physical quantities involved are subject to variability in the electrical output, and their values are determined by a system of coupled non-linear equations (*power flow equations*). When operations are applied locally, the non-linear effects propagate globally across the entire network. This incurs a substantial computational burden and calls for extensions to be developed to enable

the application of planning techniques.

We work with the planner POPF2 (Coles et al., 2010, 2011), which is a partially-ordered forward temporal planner that adopts the forward state space heuristic search framework. Forward state space heuristic search is one of the most common techniques deployed to tackle planning problems. Improving the heuristic function is a crucial part for the success of planning. In general, heuristics are based on relaxed problems: simpler problems obtained from the original by making simplifying assumptions or relaxing constraints. One of the most commonly used heuristics is the *delete list* relaxation (Bonet and Geffner, 2001). However, as pointed out in (Keyder et al., 2012), the over-simplification of the delete-relaxation can lead to uninformative heuristics. In this thesis, we present different heuristic enhancements made with the aim of improving the performance of POPF2.

1.1 Identification and Characterisation of New Challenges for Planning

In modelling the autonomous management of an electricity network we have encountered new challenges for metric temporal planning. Electricity networks are composed of various elements, such as lines, generators, transformers, loads and busbars, which are metal connecting components, linking elements of the electrical network. To describe the behaviour of electricity networks, several numeric quantities are adopted, linked together by physical laws. The most important numeric quantities are voltages and phase angles on the busbars, power and current, which vary through time as sinusoidal waves (hence the name *Alternating Current, AC*). These quantities are related to each other by the *power flow equations*, a set of coupled non-linear equations. The power flow equations are used to model the change of these quantities when transformers are stepped up and down, generators are ramped up and down, loads are shed and other actions. Typically, these actions take time to achieve their full effects and planning can be used to choose and coordinate them to achieve desired effects in the network over time.

In this thesis we are interested in the AC voltage control problem, which is one aspect of the autonomous management of an electricity network. In the AC voltage control problem, we want to maintain the voltages at all of the network busbars within a safe range over the entire period of the plan.

The domain is characterised by local actions (such as stepping up a transformer) that have global and non-linear numeric effects. Furthermore, some quantities, such as the demand and the supply, vary over time independently from all the actions taken, thus planning must take place against these background events. Historical data and predictions can be used to model these curves, which can be represented as discretised sequences of timed events, specified in the initial state. Each one of these events describes how the load and generation changes at a time-point. The presence of *trajectory constraints*, such as the limits on the voltages maintained over the entire plan, is another important characteristic of this domain.

Summarising, the key components of the AC voltage control problem are: the presence of background events that constrain the behaviour of the planner, the presence of trajectory

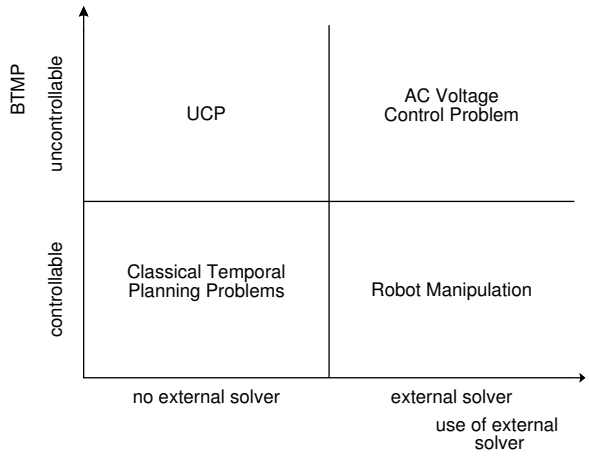


Figure 1.1: The two-dimensional space created by BTMPs and problems requiring external solver integration

constraints, some of which must hold over the entire plan, the need for temporal coordination and the global impact of the local effects of actions. This collection of features leads to the identification of two new classes of interesting temporal and numeric planning problems. The first class we call bounded trajectory management problems while the second exploits an external solver coupled with a planner to handle semantic attachments. In this thesis we explore these two orthogonal dimensions of temporal numeric planning problems, depicted in Figure 1.1, where an instance of the combination of both of them is the AC voltage control problem in distribution networks.

Bounded Trajectory Management Problems

The first class of problems that we consider is characterised by the presence of known numeric exogenous events that influence the trajectory constraints. In addition to the voltage control problem, another example of this problem is the unit commitment problem (UCP) (Campion et al., 2013), for which planning can be used to decide when to switch on and off generator units in order to satisfy a given demand. In this case, the prescheduled events are the consumers' demand, that can be assumed to be known in advance, which must always be less than the generated power controlled by the planner.

The presence of predictable numeric exogenous events is a feature not much studied in the planning community, inasmuch as they are not fully formalised in the PDDL language. A key characteristic of these numeric events is that they might introduce negative effects that are ignored in a fully delete-relaxation heuristic. In particular, we consider the impact of the uncontrollable events, represented by Timed Initial Fluents (TIFs), in problems where some metric constraints need to be maintained over an extended period of time.

Semantic Attachments

The second class of problems is characterised by the presence of more complex numeric effects (*semantic attachments*) than the ones expressible and solved in planning. The PDDL language allows only algebraic expressions between numeric fluents, and this is not enough

to model complex effects involving simple analytic functions or matrix calculations. A number of examples are present in the planning literature, such as the Space Telescope Slew Manoeuvre (Löhr et al., 2012), where a space telescope needs to rotate and zoom on different objects. The numeric quantities involved, such as the angular momentum, the angular rate and the external torque are regulated by differential bi-dimensional equations. Another real-world example is the Machine Tool Calibration domain (Parkinson et al., 2014), where planning is used to reduce the downtime and the uncertainties of measurements of machine tool calibration. The uncertainties coming from different sources need to be combined together with the square root of the sum of the squares. The Quadcopters for Surveillance Missions in (Bernardini et al., 2014) is another domain example where a non-linear time-dependent function needs to be represented. In this domain, planning is used to choose a sequence of search patterns to be executed in a Search and Track mission. Each search pattern is characterised by a reward function which corresponds to the expectation of finding a target in a search area, and its value is modelled as a Gaussian distribution among the time-points in which the target is plausibly in the search area considered.

An approach explored in (Dornhege et al., 2009), in the Robot Manipulation Problem, is to extend the planner providing external libraries to calculate complex effects and preconditions of an action and compute values of the variables. The main problem, however, is how to consider these numeric effects at the heuristic level. At the moment the most common heuristic used to treat numeric quantities is the metric FF heuristic, h^{mFF}, where only positive effects are considered in the relaxed plan graph phase. The challenge is to abstract the effects of numeric quantities calculated by the external libraries so that information can be exploited in the heuristic.

1.2 Contributions of the Thesis

The key contributions of the thesis are to:

- introduce a new class of metric temporal planning problems, in which the effects of actions interact with background events and trajectory constraints;
- develop a novel, informative heuristic method that takes background events into account;
- extend POPF2 into POPF-TIF, by replacing the Temporal Relaxed Planning Graph (TRPG) with the new heuristic method to better handle the Timed Initial Fluents (TIFs);
- introduce a general framework to integrate external evaluated functions by means of an encapsulated type and an interface with the planner;
- explore a plan-based voltage control approach for active distribution networks, and investigate the scalability of its features.

1.3 Methodology

This thesis started as an investigation into the use of planning techniques to tackle problems involving electricity networks. From this problem we have extracted two classes of numeric temporal problems which challenge the state of the art planners. We have addressed these problems by extending the modelling language and devising new algorithms for extracting heuristic values and new search strategies. The new algorithms are implemented in the planner POPF-TIF, which is an extension of the existing planner POPF2.

The algorithms are compared to the standard algorithm of the planner POPF2, through a comparative, empirical study with a benchmark consisting of different problems and domains. The performance of the new algorithms is assessed using the number of states explored with the new algorithms and the CPU time spent to solve the problems.

Another important objective of this thesis is to apply the planning technique to the AC voltage control problem and verify the benefit of the planning approach with respect to the traditional responsive approaches. For this, we have conducted a case study with a real network and we have compared POPF-TIF with an advance responsive technique. The quality of the plans produced with these two approaches is assessed by measuring a metric function over a variety of cases with and without uncertainties in the initial state.

1.4 Map of Contents

The rest of the thesis is organised as follows.

In Chapter 2 we introduce the AC voltage control problem that we are tackling with planning and that motivates our research. The chapter begins with an overview of the fundamentals of power systems, with emphasis on the voltage control problem. A section describing the future challenges of power systems gives examples that motivate the investigation of the use of planning in this field. The chapter concludes with an overview of related work about applications of AI methods in the area of power systems.

In Chapter 3 a general background to the theory behind planning is described. In the first section, we present the formalism adopted in the rest of the thesis, introducing numeric temporal planning problems and describing the state of the art techniques used in this context. Some of these techniques are borrowed from classical planning and extended to handle numeric resources and temporal constraints. The chapter includes an overview of the most recent numeric temporal planners and a detailed description of the planner POPF2, which we chose as our base planner to extend with new heuristics and search algorithms.

In Chapter 4 we introduce a new class of problem, called *Bounded Trajectory Management Problems*, in which the AC voltage control problem falls. We formally define this class in a domain independent way and we then explain the challenges that it introduces. We explain the two orthogonal approaches we explored as solutions to BTMPs, which are to improve the heuristic and to modify the search strategy. Our new heuristic is called the *lookahead heuristic*; we also consider two alternative search strategies that recognise the structure of the BTMPs to make different backtrack choices.

Chapter 5 describes the coupling of a declarative planner, POPF-TIF, with an external solver that handles non-linear numeric effects. The chapter starts with a formal description of

the semantic attachment that we are including in our approach and a comparison with other forms of semantic attachments. We then describe the extension of the modelling language to represent problems with external evaluated functions, obtained with the introduction of *encapsulated types*. Next, we explain the implementation of the interface between the planner POPF-TIF with external modules. The chapter concludes with a discussion on ways to abstract the effects calculated with external solvers. This abstraction is used at the heuristic level to approximate these effects.

In Chapter 6 the PDDL model developed to represent the AC voltage control problem is presented. We explain different approaches that can be used to model network operations and we report the final domain model. We also provide the details for generating problem files and the abstractions used in this domain. The last section of the chapter describes the specific external solver used and a comparison with other alternatives.

Chapter 7 contains the experimental evaluation of our approach and it is divided into two parts. The first part describes the results of the AC voltage control problem applied to different distribution networks. In particular we compare our approach with a modern reactive control strategy that is used in industrial power systems management and we examine the scalability of our approach. The second part of the chapter is dedicated to an empirical study of the *lookahead* heuristic and the alternative search strategies applied to different BTMP domains.

Chapter 8 concludes the thesis with a summary of the material presented and a synthesis of what can be drawn from the thesis. We also identify possible directions of further research.

1.5 Publications

Chiara Piacentini, Varvara Alimisis, Maria Fox, Derek Long. 'Combining a Temporal Planner with an External Solver for the Power Balancing Problem in an Electricity Network'. In *International Conference on Automated Planning and Scheduling*, 2013. The electricity network balancing problem consists of ensuring that the electricity demands of the consumers are met by the committed supply. Constraints are imposed on the different elements of the network, so that damage to the equipment is prevented when transformers are stepped up or down, or generation is increased. We consider this problem within zones, which are sub-networks constructed using carefully chosen decomposition principles. The automation of decision making in electricity networks is a step forward in their management which is necessary for coping with the increase in power system complexity that we expect in the near term. In this paper we explore the deployment of planning techniques to solve the zone-balancing problem. Embedding electricity networks in a domain description presents new challenges for planning. The key point is that the propagation of information requires complex updates to the state when an action is applied. We have developed a method in which the computation of the critical numeric quantities is performed calling an external power flow equation solver, demonstrating a clean interface between the planner and this domain-specific computation. This solver allows us to move the power flow computations outside of the planning process and update the values efficiently. We also examine a second important feature of this problem, which is the interaction between

exogenous events and constraints over the entire plan trajectory within a zone.

Chiara Piacentini, Maria Fox, Derek Long. 'Planning with Numeric Timed Initial Fluents'. In *AAAI Conference*, 2015. Numeric Timed Initial Fluents represent a new feature in PDDL that extends the concept of Timed Initial Literals to numeric fluents. They are particularly useful to model independent functions that change through time and influence the actions to be applied. Although they are very useful to model real world problems, they are not systematically defined in the family of PDDL languages and they are not implemented in any generic PDDL planner, except for POPF2 and UPMurphi. In this paper we present an extension of the planner POPF2 (POPF-TIF) to handle problems with numeric Timed Initial Fluents. We propose and evaluate two contributions: the first is based on improvements of the heuristic evaluation, while the second considers alternative search algorithms based on a mixture of Enforced Hill Climbing and Best First Search.

Chiara Piacentini, Varvara Alimisis, Maria Fox, Derek Long. 'An Extension of Metric Temporal Planning with Application to AC Voltage Control'. Accepted for publication in *Artificial Intelligence Journal*. In this paper we explore the deployment of planning techniques to solve a new class of metric temporal planning problems, characterised by the need to manage both plan trajectory constraints and uncontrollable numeric events. This combination gives rise to challenges not previously solved in state-of-the-art planners. We introduce new planning methods to handle these challenges, and demonstrate our approach using a real application domain: voltage control in Alternating Current (AC) electrical networks. Embedding electricity networks in a domain description presents important modelling challenges. We introduce an encapsulated type, Network, the implementation of which is hidden from the planner. The effects of actions trigger complex updates to the state of the network. We distinguish between the direct effects of planned actions, and the indirect effects triggered by them, and we propose a method for integrating a specialised external AC power equation solver with a planner. We consider a new heuristic function that takes into account the next uncontrollable event, and its interaction with active trajectory constraints, when determining the actions that are helpful in a state. This lookahead heuristic also exploits an abstraction of the encapsulated Network type to obtain more informative distance estimates. We conduct experiments to evaluate the benefits of the lookahead heuristic, showing that our approach scales very well with the size of the network and the number of controllable components of the network.

1.5.1 Other Publications

Varvara Alimisis, Chiara Piacentini, James E. King, Philip C. Tylor. 'Operation and Control Zones for future complex Power System'. In *IEEE Green Technologies Conference Operation*, 2013. This paper presents zones for the control and operation of future complex power systems, which could be highly distributed both in terms of generation sources but also concerning the control and decision making components. The zones presented are sub regions of a power network and constitute sympathetic entities in terms of operational flexibility, with potentially dynamic boundaries, serving a judiciously

selected control algorithm or technique. The IEEE 57-bus power system is used as a case study and the significance of the control intelligence to zoning is further emphasized.

Varvara Alimisis, Chiara Piacentini, Philip C. Tylor. 'Zoning reconfiguration for Coordinated Voltage Regulation in Future Transmission Grids'. In *6th ISGT North America, IEEE Conference*, 2015. Hierarchically structured Automatic Voltage Control (AVC) architecture enables wide-area closed-loop Coordinated Voltage Regulation (CVR). Owing to the inherent complexity of the task, CVR relies on reduced control models, i.e. simplified models of the system suitable for voltage control. It is a fact however that a single reduced control model (static RCM) cannot be optimal for all network configurations and operating conditions. In pursuit of advanced online voltage control for a future smart transmission grid, this paper presents adaptive control model reduction (adaptive RCM) for CVR. The proposed formulation is based on a complex network representation of the transmission grid. Additionally, the formulation employs spectral clustering complemented by perturbation theory to deliver the adaptive RCM. Indicative results are presented on the New England 39-bus network and provide adequate justification of the proposed approach. The implications of an adaptive RCM scheme to control algorithm design and selection are also discussed.

Chapter 2

Motivation

In this chapter we present the fundamentals of power systems and we describe the drivers that will shape the future power systems and will challenge the current infrastructure. These justify an investigation of the application of planning to power systems. In particular, we consider the voltage control problem, which is described in detail, followed by an overview of related work in AI.

2.1 Power Systems

Power Systems are nationwide infrastructures that supply electrical energy to consumers over large geographical areas in real time (Machowski et al., 2008). Before reaching the consumers, the electricity needs to be generated, transmitted and distributed. Decisions about which generators to use, over which time periods and at what levels of output, are determined centrally, in line with predicted consumer demand, market forces and other pressures.

In the future, fundamental changes are anticipated, as a consequence of environmental, political and economic considerations, global warming, population and load growth and increased dependence on imports. A shift in technology and operational philosophy is expected, in response to an increased penetration of renewables, the electrification of transport and heating and a new role for the consumer, who will actively participate in managing the network.

2.1.1 Power Systems Background

A power system comprises several distinct parts, as schematically represented in Figure 2.1:

- Generation units: the traditional generation process is based on a relatively small number of thermal or hydro plants that convert thermal or mechanical energy into electricity. The electricity produced by these units is then stepped-up to high voltage and fed to the transmission network.

- Transmission network: the electrical energy produced in power plants is transmitted over long distances at high voltage through the transmission network. Increasing the

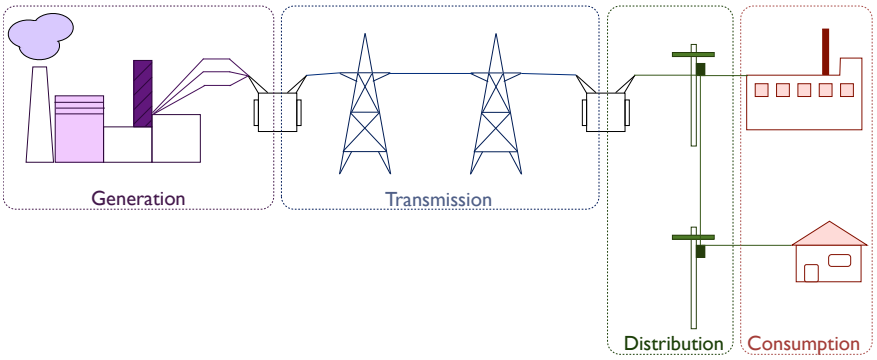


Figure 2.1: Schematic of a typical Power System.

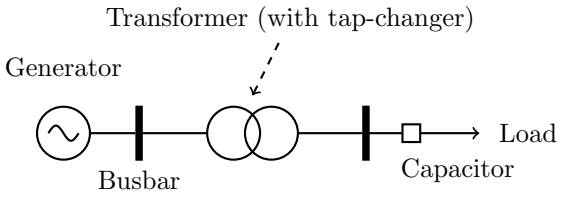


Figure 2.2: A fragment of an electrical network, showing two busbars, a generator, a transformer and a capacitor.

voltage reduces transportation losses, since the losses are proportional to the square of the current and current is inversely proportional to voltage at a given power level.

- Distribution network: from the transmission network the electrical energy is transferred to high-voltage and medium-voltage distribution networks and it is then delivered to the consumers. A fragment of a distribution network is shown in Figure 2.2. It is composed by two busbars, metallic pieces that conduct electrical current, a generator, a transformer and a capacitor.

- Consumers: the electrical energy consumption is the electrical energy absorbed by the final user. Consumers can be residential, commercial or industrial, each connected to the appropriate voltage level.

By contrast with other infrastructures, such as water and gas, power systems have to meet the energy demand with the generators at any instant, because electrical energy cannot be easily and conveniently stored in large quantities. Although individual consumer demand is unpredictable, the combination of various load patterns follows a more smooth and predictable trend, as shown in Figure 2.3. This property makes it possible to use automated planning for the control of the operations of the network given an average load profile. In current power systems, generation can also be treated as predictable so that, over a 24-hour period, it can be assumed that both load and generation profiles are known. The transmission and distribution networks are constrained to maintain given standards to guarantee an efficient transportation, safety and usability of the electrical current, as well as guarantee high reliability of the supply.

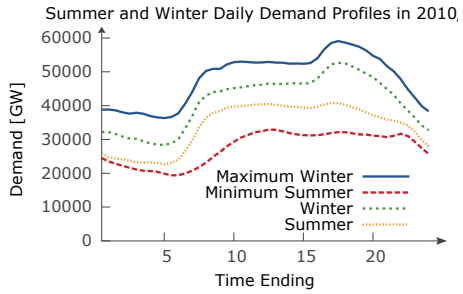


Figure 2.3: UK loads profiles for typical summer and winter days and minimum and maximum loads for years 2010/2011 (Copyright, National Grid PLC 2010).

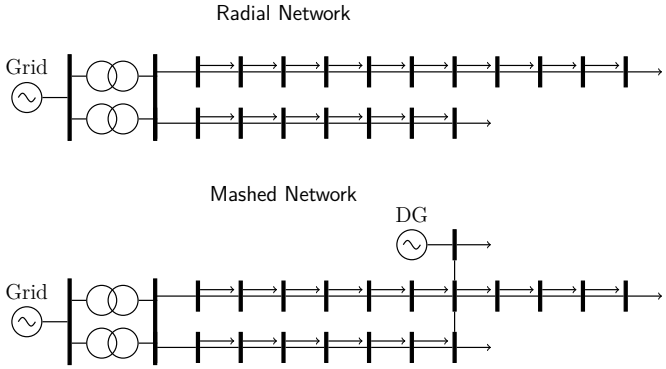


Figure 2.4: Radial and meshed network topologies.

Network topology

A *radial network* is a tree-structured network with a single generation source at the root. All of the network structure "radiates" from this single source. The main branches of this radial structure are referred to as *feeders*. Consumers (loads) are at the leaves of the tree. A *meshed* network can contain redundant lines between busbars so that the generations unit are interconnected. When there is ad hoc generation attached at the distribution level (for example, when consumers also participate in electricity generation), a meshed network results. Unless islanded, such a network remains attached to the grid. Figure 2.4 shows two examples of radial and meshed network topologies.

2.1.2 The Future of Power Systems

Power Systems are gradually evolving through time. Since 1891, with the opening of the Deptford Power Station in London, the first modern high-voltage power station, the consumption of electricity has shown a continuous growth (Figure 2.5). Electrical generation has been subject to various changes: from the beginning of the 1920s, UK electrical generation relied on coal. Although nowadays coal is still the main resource used to produce electrical energy, from the 1950s oil started to have a significant role, overtaken in the mid 1980s by natural gas. In the most recent years, the use of distributed (or embedded) generators (DG) connected to the distribution networks, as wind or solar plants, has shown a high growth trend helped by the improvement in performance of small plants, the incentive

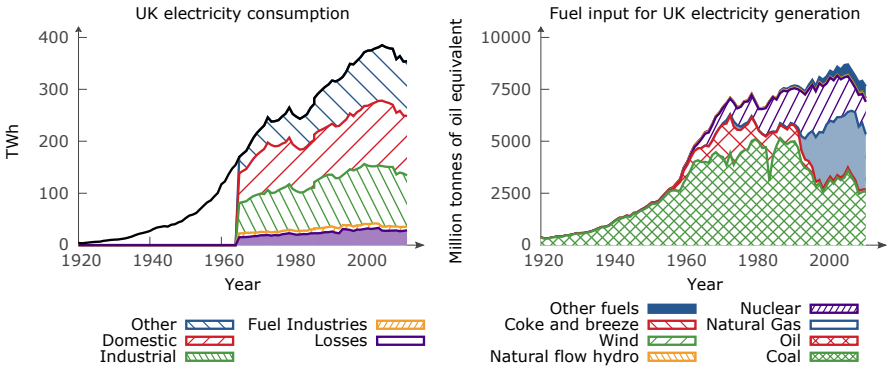


Figure 2.5: UK percentage of fuel input for electricity generation (Government Digital Service, 2014).

schemes for the exploitation of renewable resources and the electrical system liberalisation contributed to the diffusion of distributed generators (Bignucolo et al., 2008).

In the future fundamental changes are anticipated. The UK in particular has committed to reduce its greenhouse gas emissions by at least 80% by 2050, relative to 1990 levels (Fardanesh, 2002; Department for Business Enterprise & Regulatory Reform, 2008). To support this transition, several changes are expected; increased penetration of renewables, distribution networks transforming from passive to active, electrification of transport and heating and a new role for the consumer-prosumer who would actively participate in managing the network. All these drivers will reshape the energy use and electrogeneration mixture and would further add more operational complexity to the power system.

In order to deal with the increase in complexity, one technique that can be adopted is to decompose the entire power network into smaller *zones* (Alimisis and Taylor, 2014). A zone is a physically connected collection of power elements, chosen to be strongly self-contained in terms of the effects of actions applied to those elements, and largely unaffected by the effects of the same control actions applied to elements outside the zone boundaries. The reduced complexity of the zones allows the application of automated planning, and other pro-active techniques, to manage the operations inside them, considering temporal and metric constraints.

2.2 Voltage Control in Distribution Networks

In order to deliver good quality electrical supply, an important parameter to be considered is the voltage level, that should not fluctuate much from the nominal value. In the UK, the voltage level to be maintained is established in the Electricity Safety, Quality and Continuity Regulations (Stationery Office, 2002). Distribution networks supply electricity at 132 kV and below. Technical and economic considerations impose certain bounds on voltage nominal values. In networks operating below 132 kV, voltages can exceed their nominal values by 6 per cent, while for networks operating above 132 kV, the tolerance limit is 10 per cent. The low voltage supply is set to be at 230 volts, with a maximum variation of 10 per cent above and 6 per cent below the declared voltage at the declared frequency.

2.2.1 Theory of the Voltage Control Problem

In an AC power system, current and voltage follow sinusoidal waves. The root mean square of the product of current and voltage gives the *apparent power* flow on the network. The difference between the origins of the current and voltage curves is called the phase angle, giving the complex number (r, θ) where θ is the phase angle and r is the magnitude of the apparent power. The real and imaginary parts of this number correspond to the real and reactive power components, referred to as P and Q respectively. The reactive power Q refers to the maximum value of the instantaneous power absorbed by the reactive component of the load (such as refrigerators, air conditioners, clothes dryers, and other devices with electric motors). Resistive load, such as heating elements, absorb only real power, P.

The AC voltage control problem is concerned with maintaining the voltage at each busbar of a distribution network (or a zone of the network), whether meshed or radial, within a given bound, for an extended period of time, assuming that the load profile of consumers and the schedule of the generators are known throughout this time. The voltage at each point of the power system is influenced by different factors: the topology of the network, the load consumption and the power generated. The decision variables are the tap ratios of transformers, the status of the capacitors and reactors and the level of power demand that can be shed from the network, while the network has to respect the voltage constraint:

$$V_{k,min} \leq V_k \leq V_{k,max} \quad \forall k = 1, ..., N \tag{2.1}$$

where N is the total number of busbars present in the network. The voltage V_k at busbar k can be obtained using the *power flow* equations:

$$\begin{aligned} P_k &= \mathrm{V_k} \sum_{n=1}^{N} \mathrm{V_n} \left[G_{kn} \cos\left(\delta_k - \delta_n\right) + B_{kn} \sin\left(\delta_k - \delta_n\right) \right] \\ Q_k &= \mathrm{V_k} \sum_{n=1}^{N} \mathrm{V_n} \left[G_{kn} \sin\left(\delta_k - \delta_n\right) - B_{kn} \cos\left(\delta_k - \delta_n\right) \right] \end{aligned} \tag{2.2}$$

These are a set of non-linear equations where P_k and Q_k are respectively the real and the reactive power injected/absorbed in the busbar k, δ_k is the phase angle of the alternating voltage, and G_{kn} and B_{kn} are called, respectively, the reactance and susceptance of the line between busbars k and n. Further details of the power flow equations can be found in Appendix A.

Due to the non-linearity of the power flow equations, their integration through traditional decision-support tools is very limited. Instead, different linear approximations can be applied in situations in which it is possible to consider only the active power. This is the case of transmission networks, where the ratio G_{kn}/B_{kn} is very small, the phase angle difference is small enough and the voltage magnitudes are close to 1.0 pu[1], so that the Equations 2.2 reduce to:

$$P_k = \sum_{n=1}^{N} -B_{kn}(\delta_k - \delta_n) \tag{2.3}$$

[1] In power systems the *per-unit* system is adopted. For a given quantity, in this case the voltage magnitude, the *per-unit* value is obtained by scaling the quantity to a reference value.

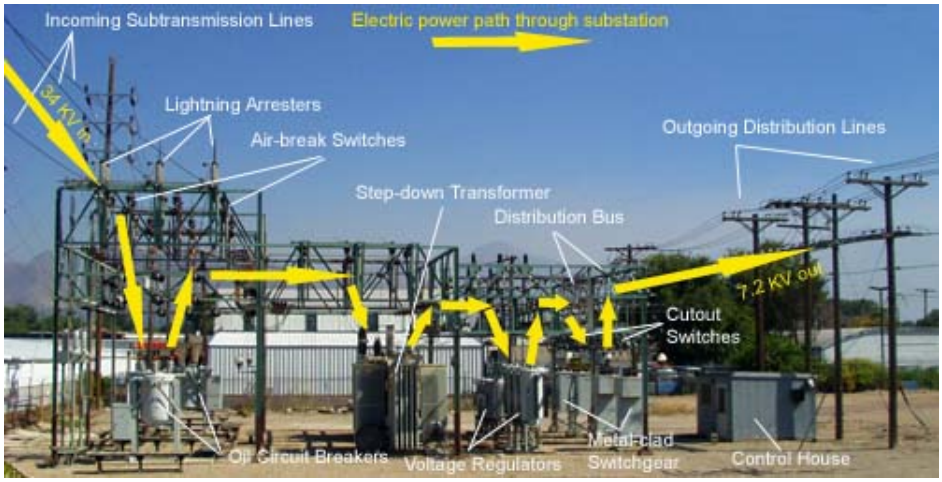


Figure 2.6: Energy flow through a typical substation. Credit-OSHA.

A recent review of the different approximation, called DC power flow, can be found in (Stott et al., 2009). The approximations are different in terms of the definition of the admittance matrix and the injection in the power flow equation. At the distribution level, however, the voltage ratio between the conductance and the suscepectance is not negligible, and the reactive power needs to be taken into account in the computation of the voltage variables.

2.2.2 Reactive Voltage Control

Generally, the equipment used for the voltage control in transmission and distribution networks is located in *substations* such as the one in Figure 2.6. The substations provide system security via automatic protective devices, control the flow of electrical current and transform the voltage from one level to another.

In the following the devices used to provide voltage control are described:

- Transformers are used to control the voltage of the power flowing into them. There are different types of transformers: step-up transformers, typically found in generation plants, used to step up the voltage from low to the high transmission voltage levels; step-down transformers, used to step voltage down to levels appropriate for consumers, and regulating transformers, used to maintain proper distribution voltage. All the transformers have the same working principles: they are composed of two coils of wire and a voltage source. The current flowing in the coil on one side of the transformer induces a voltage in the coil on the other side. The voltage on the opposite side changes directly proportionally to the ratio of the number of turns in the transformer. The regulation of the ratio of turns is realised by a *tap*.

- Capacitors have the property of delaying the phase of the alternating current with respect to the alternating voltage. For this reason they can be considered as sources of reactive power. In substations, capacitors are connected in capacitor banks that can be switched on or off to reduce system losses and to provide voltage support.

- Reactors are high-voltage inductors and they delay the voltage with respect to the electric current. They can be considered as absorbers of reactive power. Reactors are used to help with the regulation of the transmission system voltage, absorbing the excess reactive power or reducing fault current in distribution lines.

- Static VAR Compensators (SVC) are composed of several capacitors and inductors (reactors) and a switching system that can inject or absorb reactive power in the power system, helping in the regulation of the voltage at their terminals.

The network operations are controlled by Supervisory Control and Data Acquisition (SCADA) systems. A SCADA system is composed of a central computer connected to several remote terminal units (RTUs) located throughout the system, which collect network operational states by mean of sensors. The data are sent to the central unit every 10/20 seconds for higher voltage networks or on an hourly or daily basis for lower voltage sites. The control centre detects anomalous conditions and sends to the remote control elements signals to operate corrective actions (Yang et al., 2011).

In distribution networks with a typical radial configuration, voltage control strategies are based on changing the settings of transformer taps and the local regulation of capacitor banks connected along the feeder. The penetration of distributed generators has an impact on the voltage profile along the feeder, because they inject active and reactive power and change the configuration of flows in the network. This means that the voltage does not drop along the feeder as in the current passive network, creating the need for different voltage control mechanisms. Several solutions are presented in the literature (Roberts et al., 2003; Davidson et al., 2009; Mutale, 2006; Peikherfeh et al., 2011; Zhao et al., 2014), which range from a reconfiguration of the network (changing its topology from radial to meshed and requiring massive investment) to proactive management of the elements in the distribution network. Automated planning provides a means by which proactive management can be achieved.

The Hierarchical Tap Changing Method

An advanced method in AC voltage control is the Hierarchical Tap Changing (HTC) approach (Larsson, 2000). This is a reactive control[2] approach with indirect coordination without the need for communications. The structure of the hierarchy determines the order in which transformers respond to changes in the voltages at the busbars. Power normally flows downstream from the main source of generation. Each transformer maintains the voltage of its adjacent downstream busbar within the specified bounds by adjusting its tap ratio according to the following equation:

$$t_{new} = \frac{V_{e,target}}{V_{e,measured}} t_{old}, \tag{2.4}$$

where t_{new} and t_{old} are the tap ratios of the transformer, before and after the operation, and $V_{e,target}$ and $V_{e,measured}$ are the nominal voltage and the actual measured voltage at

[2]By "reactive" here, we mean that the control method reacts to control inputs, rather than that it controls reactive power

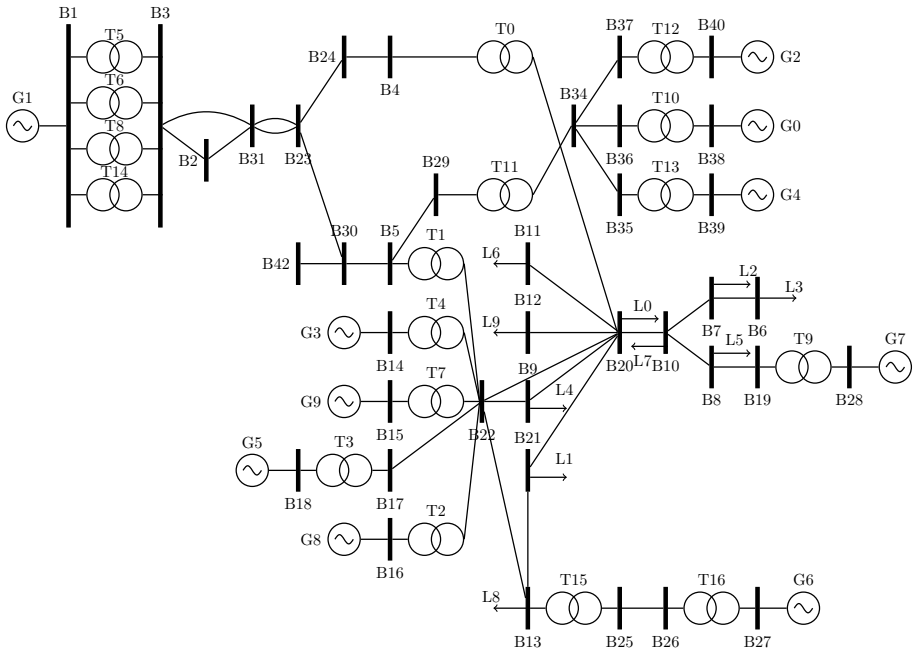


Figure 2.7: The 33 kV interconnected AuRA-NMS network.

the downstream busbar for the transformer. The indirect coordination is achieved through the use of fixed time delays in the order of one minute (Larsson, 2000).

The order in which the transformers in this hierarchy are triggered is determined by time delays which are longer for transformers at the bottom of the hierarchy than for those at the top. Although the HTC approach is a modern and sophisticated technique, it can be challenging to coordinate activity using simple fixed time delays on the transformers.

2.2.3 The 33 kV AuRA-NMS Network

In this thesis we consider as benchmark network the 33 kV interconnected network taken from Davidson *et al.* (Davidson et al., 2010) and shown in Figure 2.7. This network was developed as part of the AuRA-NMS (autonomous regional active network management system) project, whose aim was to design distribution networks where distributed and renewable generators could connect into the network without large cost penalties. The 33 kV network contains 39 busbars, 9 distributed generators and 17 transformers. The substation (B1 and B3) connects the 33 kV network to the rest of the grid represented by the generator G1. In contrast to a traditional radial distribution, where the busbars are usually located in sequence, the topology of this network presents a *meshed* configuration. This network has now become a benchmark network for studies of future power systems (Davidson et al., 2009; McArthur et al., 2012; Hu and Li, 2012; Kings et al., 2014) because it both represents a real physical network and incorporates distributed generation.

Amongst the studies on this network, the work of Davidson *et al.* (2009) presents a formulation of the active management of the networks at given time-points. The problem is formulated as a constraint satisfaction problem (CSP), where the control variables determine

the outputs of a set of generators, which have discretised domains. Contractual and preference constraints are taken into account. Although this approach produces feasible solutions for quite small problems, for larger networks the combinatorial size of the network and the absence of a heuristic evaluation to guide the search limit the possible applications. Moreover, the authors consider single instants of time and temporal constraints are not addressed.

2.3 Related Work in Power Systems

Automated planning has already been applied to problems involving distribution networks. For example, Bell *et al.* (2009), present an application of planning to managing voltage control in a power station. Restricting attention to a power substation, without the presence of generators, and considering only a small portion of the distribution network, allowed some simplifications of the effects of the other numeric quantities on the voltage. In practice, the constraints imposed by the Equations 2.2 are not considered, while it is assumed that each variation in consumed power leads to a constant effect on the voltage. In the work of Cao *et al.* (2011) an entire distribution network is considered. In this case the power flow equations cannot be ignored, but they are not implemented directly in the planner. The authors adopt a linearisation of the effects of power flow in the planning model and only after the planner finds a solution, is it verified against the power flow equations. In case the solution is not valid, the model is modified with refined values and re-solved.

Another example of the use of automated planning in Power Systems is the Power Supply and Restoration (PSR) domain (Thiébaux and Cordier, 2001). In that work the authors consider a distribution network and the objective is to maximise the load connected to the generators without passing through a faulty line. In that case the power flow equations are not considered and the domain is expressed entirely using propositional variables.

More recent work on the PSR problem, by Thiébaux *et al.* (2013), formulates the problem as a mixed integer program (MIP) exploiting the use of a Direct Current (DC) simplification of the power flow equations.

Work on optimising power flow in microgrids, which considers the power flow equations, can be found in a later work of Scott *et al.* (2014). In this work the problem is formulated as a MIP, exploiting better approximations than the DC simplification. To optimise the efficiency of the network, they use the Alternating Direction Method of Multipliers (ADMM), which is a refinement of the use of Lagrangian multipliers. This approach was also used for a similar purpose by Phan and Kalagnanam (2014). The resulting optimisation problem is solved using a combination of linear and non-linear solution methods. In these problem formulations, voltage is not controllable but instead imposes constraints on the power flow.

Work by Tischer *et al.* (2011) addresses the problem of demand response within a power flow model. This impacts on voltage control, but is not the same problem. Consumers are modelled as participants who will change their own behaviour in response to financial and comfort considerations. Real time pricing signals are used to drive consumer behaviour towards optimal power flow distributions.

Coffrin *et al.* introduce a more sophisticated approximation of the power flow equations using a Linear Programming Approximation of the AC Equations (Coffrin and Van

Hentenryck, 2012). Here, the non-linearity of the AC power flow equations is overcome by using a piecewise-linear approximation of the cosine. In this formulation, the values of active and reactive power can be decision variables and the model can be embedded in a MIP solver for making discrete decisions about the power system. The weakness of this formulation comes from the non-linear behaviour of transformers, that introduces a loss of accuracy in the power flow calculations.

A CSP approach for the active management of the networks is reported in the previous section. While CSP and SAT-based approaches, including CSP-based planning (Vidal, 2011), are typically based on the discretisation of time, planning can exploit continuous time. In planning, once the action is selected, the decision variable is the time at which it should be applied. Although in our model we discretise the functions that represent both load and supply, and describe them by giving values at specific time-points, using Timed Initial Fluents (TIFs), we do not discretise time, and actions can be applied whenever the planner chooses.

Chapter 3

Automated Planning

In this chapter a review of temporal numeric planning is presented. Starting from the definition of the temporal numeric planning problems and the modelling languages, we discuss the main approaches applied to extend classical planning to handle numeric resources and temporal aspects. We conclude with an overview of the *state of the art* numeric temporal planners, focusing on POPF2.

3.1 Planning Formalism

Automated planning is a branch of Artificial Intelligence concerning the finding of an ordered sequence of actions that achieves a goal from an initial state. Research in planning branches into various directions, depending on the assumptions that are made about the world and the kinds of transitions modelled between states. In this thesis, we are interested in particular in temporal planning, because it allows an explicit representation of time and consequently an explicit representation of temporal constraints (Figure 3.1). In this section, we describe the general framework that we use in this thesis, focusing on the formalisation of the metric temporal planning problem.

3.1.1 The Metric Temporal Planning Problem

We formalise the metric temporal planning problem introducing the concepts of state and action.

Definition 1 (State) *Given a set of literals L and a set of numeric variables V, a* state *in planning is an assignment of values to the literals in L and to the numeric variables in V.*

Definition 2 (Durative action) *A durative action* a *is a 6-tuple* $\langle dur, pre_{\vdash}, eff_{\vdash}, pre_{\leftrightarrow}, pre_{\dashv}, eff_{\dashv} \rangle$*, where*

- *dur are the duration constraints, expressed as a conjunction of metric constraints of the form*

$$t_{\dashv} - t_{\vdash} \bowtie c,$$

where $t_{\dashv}$ *and* $t_{\vdash}$ *are the times at which* a *ends and starts respectively, c is a constant number and* $\bowtie \in \{<, >, \leq, \geq, =\}$*;*

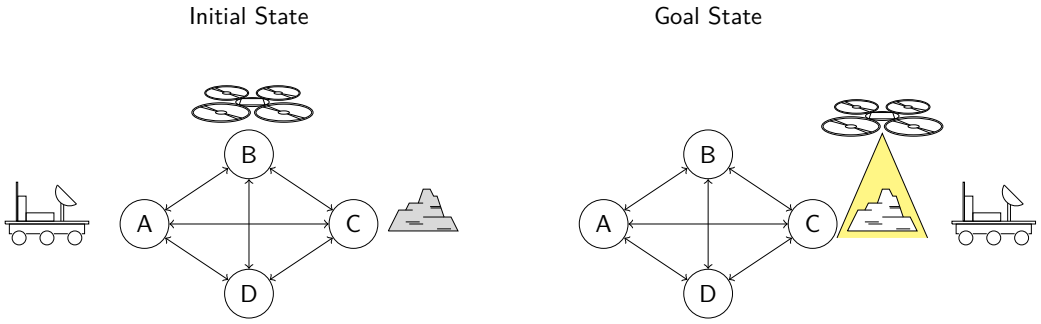


Figure 3.1: Example of temporal planning task. A rover explores the surface of a planet, but it needs the light of a drone to reach shadowed rocks.

- $pre_{\vdash}$ *(*$pre_{\dashv}$*) are the logical sentences over the literals in* L *and the comparisons of metric expressions over the variables in* V *that must hold at the start (end) of the action;*

- $pre_{\leftrightarrow}$ *are the logical sentences over the literals in* L *and the comparisons of metric expressions over the variables in* V *that must hold in the open between the start and the end of the action;*

- $eff_{\vdash}$ *(*$eff_{\dashv}$*) are the start (end) effects of the actions, expressed as partial assignments of literals in* L *and numeric variables in* V*.*

Each action defines two time-points: the *start* and the *end* of the action, which we call *happenings*.

Given the definition of *state* and *action*, we can write the planning problem as follows:

Definition 3 (Metric temporal planning problem) *A* metric temporal planning problem *is a 3-tuple* $\mathcal{P} = \langle \mathcal{I}, \mathcal{G}, \mathcal{A} \rangle$*, where* $\mathcal{I}$ *is the initial state,* $\mathcal{G}$ *is a set of goal states, and* $\mathcal{A}$ *is a set of actions.*

Definition 4 (Plan) *Given a planning problem* $\mathcal{P} = \langle \mathcal{I}, \mathcal{G}, \mathcal{A} \rangle$*, a* plan π *is a list of timed actions with a given duration* (t, a, d)*, where* t *is a rational-valued time,* $a \in \mathcal{A}$ *is an action, and* d *is a rational-valued duration.*

Given a plan π containing n actions, the plan induces a sequence of at most $2 \cdot n$ states $s_{t'}$. Each of these states is generated by applying the effects ($eff_{\vdash}(a_i)$ and $eff_{\dashv}(a_i)$) of all the actions a_i that start (end) at time $t' = t$ ($t' = t + d$).

Definition 5 (Solution) *Given a planning problem* $\mathcal{P} = \langle \mathcal{I}, \mathcal{G}, \mathcal{A} \rangle$*, and a plan* $\pi = \{(t_1, a_1, d_1), (t_2, a_2, d_2), ..., (t_n, a_n, d_n)\}$*,* π *is a* solution *of the problem* $\mathcal{P}$ *if :*

- $\forall i = 1, ..., n$*, the* $pre_{\vdash}(a_i)$ *hold in the state* s_{t_i}*;*

- $\forall i = 1, ..., n$*, the* $pre_{\dashv}(a_i)$ *hold in the state* $s_{t_i + d_i}$*;*

- $\forall i = 1, ..., n$*, the* $pre_{\leftrightarrow}(a_i)$ *hold in every state* s_t*, with* $t_i < t < t_i + d_i$*;*

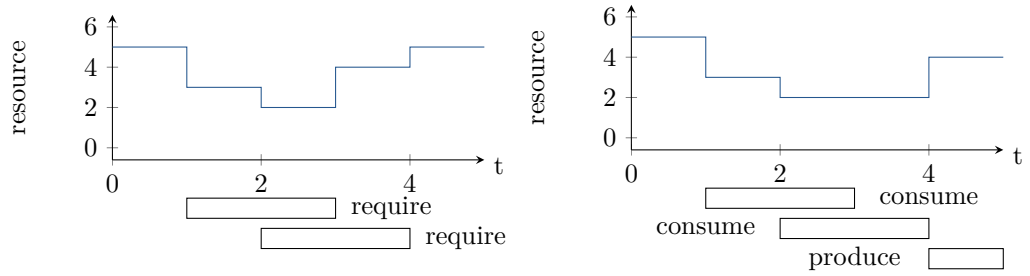


Figure 3.2: Example of consumable and reusable resources. Each box is an action, with the width representing its duration.

- $\forall i = 1, ..., n$, *the duration constraints* dur_i *are satisfied;*

- $s_{t_{max}} \in \mathcal{G}$.

With this formalism, it is possible to describe problems with temporal co-ordination, and temporal constraints. We operate under various assumptions: states of the world are fully observable and known in advance, actions are fully deterministic and the world is static, meaning that states are changed only by means of actions, while external events are not considered. We can make a simple extension of temporal planning to support a particular type of external event, which is known in advance and happens at a known time-point. We call these *predictable exogenous events*.

When we do not consider time or metric fluents, we can simplify the problem to that of **classical planning**. This is the simplest formulation of planning, where actions do not have duration and are expressed only by their preconditions (pre) and their effects (eff). Plans are expressed only as a sequential list of actions.

3.1.2 Numeric Resources

An important aspect of planning for real world problems is the use and the handling of numeric resources. Resources in general (not in strictly numerical quantities) can be seen as objects that are needed to perform an action. There are different types of resources. A first distinction can be made between *reusable* and *consumable* resources. The reusable resources are the ones that are used during the execution of the action, but after its termination they are released, as shown in left plot of Figure 3.2. On the other hand, a consumable resource is no longer available after usage, although other actions can be deployed to produce more resources. An example of consumable resource is shown in the right plot of Figure 3.2.

Another classification of numeric resources is based on how the resource varies over time. In particular we can identify discrete or continuous resources. Discrete resources change instantaneously as an effect of actions (e.g: production of bread), while continuous resources (e.g: fuel, which is continuously consumed) are generally dependent on time, and they can be modelled using effects that persist over time. The branch of planning that deals with both discrete and continuous resources is called Hybrid Planning (Fox and Long, 2006; Coles et al., 2009b; Della Penna et al., 2009; Löhr et al., 2012). The difference between discrete and continuous resources is shown in Figure 3.3.

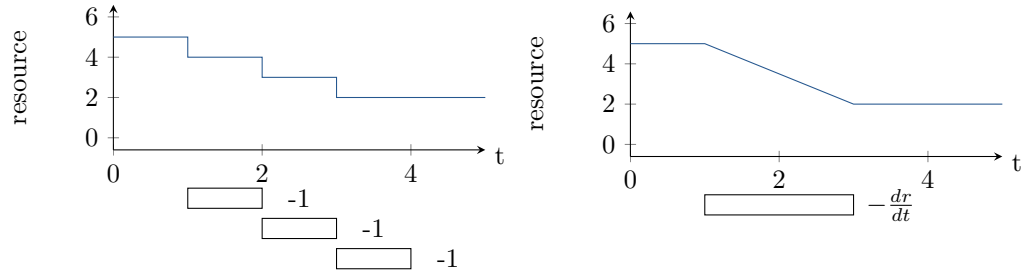


Figure 3.3: Example of discrete and continous resources. Each box is an action, with the width representing its duration.

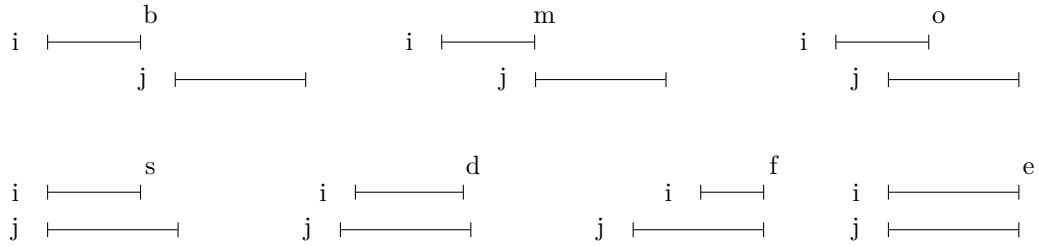


Figure 3.4: Primitive relations in Interval Algebra

3.1.3 Representation of Time

The temporal relationship between happenings can be of two different forms: *qualitative*, and *quantitative*. In the first approach, the set of instants are related qualitatively (e.g. t_1 happens before t_2), while in the second, numerical constraints are used (e.g. $t_1 - t_2 = a$, with $a \in \mathbb{R}$).

Point Algebra (PA) (Vilain and Kautz, 1986) is used to express time in a qualitative way. It relates two time-points with a qualitative constraint, without necessarily ordering them.

Similarly, Interval Algebra (IA) (Allen, 1983) represent qualitative temporal constraints, but it relates intervals of time with each other, rather than with points of time.

An interval of time i is characterised by two ends-point $i^-, i^+ \in \mathbb{R}$, with $i^- < i^+$. Given two intervals i and j, there are thirteen possible primitive relations between them, $P = \{b, m, o, s, d, b', m', s', d', f', e\}$ (Figure 3.4):

- *before (b)*, *meet (m)*, *overlap (o)*, *start (s)*, *during (d)* and *finished (f)*;
- *after (b')*, *is met by (m')*, *is overlapped by (o')*, *is started by (s')*, *includes (o')* and *is finished by (f')*;
- *equal (e)*.

According to the definition of durative actions as reported in Definition 2, the start and the end points of actions can be subject to constraints expressed as metric comparisons. The qualitative constraints expressed as PA or IA cannot be used to represent such constraints. Instead, one can use Simple Temporal Networks, which takes into account quantitative temporal constraints.

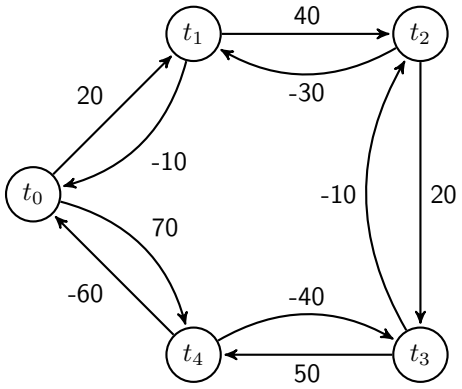


Figure 3.5: Example of STN graph

Simple Temporal Networks

Simple Temporal Networks (STN) are used to express quantitative temporal constraints (Dechter et al., 1991). If we consider a set of time-point variables $X = \{t_1, ..., t_n \mid \forall i = 1, ..., n \; t_i \in \mathbb{R}\}$, then each time-point can be subject to:

- unary constraints: $a_i \leq t_i \leq b_i$
- binary constraints: $a_{ij} \leq t_j - t_i \leq b_{ij}$

with $a_i, b_i, a_{ij}, b_{ij} \in \mathbb{R}$. Using an origin reference point t_0, one can transform the unary constraints into binary constraints of the form: $a_i \leq t_i - t_0 \leq b_i$.

Definition 6 (Simple Temporal Network) *A* Simple Temporal Network (STN) $\mathcal{T}$ *is defined as the pair* (X, C)*, where* X *is the set of time-points* $X = \{t_1, ..., t_n\}$ *and* C *is a set of binary constraints specified by a single interval. A solution of the STN is a tuple* $\{x_1, ..., x_n\}$ *if the assignment* $t_1 = x_1, ..., t_n = x_n$ *satisfies all the constraints. The STN is called* consistent *if there exists at least one solution.*

An STN can be represented as a directed edge-weighted graph $\mathcal{G} = (V, E)$, where the nodes represent variables and the edges $i \rightarrow j$ are labelled by a weight a_{ij} representing the linear inequality $t_j - t_i \leq a_{ij}$. An example containing five time-points is shown in Figure 3.5.

Using this data structure, it is possible to prove that a given STN $\mathcal{T}$ is consistent if and only if the associated graph $\mathcal{G}$ has no negative cycles.

3.2 Modelling Languages

In this thesis we adopt the PDDL2.2 planning language, which is the temporal and numeric extension of PDDL with timed initial literals. In the rest of the section we describe the PDDL family of languages, starting from PDDL and explaining the extensions we use.

3.2.1 The PDDL Language

Since the first *International Planning Competition* (IPC) in 1998, the official language became PDDL (*Planning Domain Definition Language*) (McDermott et al., 1998). Here

the closed world assumption is adopted. As stated by the authors, this language is similar to the predecessor ADL (Pednault, 1989) in expressiveness, however, since a planner can be built to support partial features of the language, an additional field is added, which allows specification of which features are required to model the problem (e.g. universal quantifiers, equalities, etc.). This gives rise to a family of PDDL languages.

In PDDL, the definition of a planning task is divided into two components:

- **domain description**: this contains the type hierarchy, constant objects, predicates, and operator schema;
- **problem description**: this contains the specific objects and their types, the initial state facts and the goal conditions.

This subdivision reflects the separation of the elements that remain unchanged with different instances of the problems and the ones that are specific to the problem instance. In this way, it is possible to connect several problem descriptions to the same domain description.

Different versions of PDDL support different aspects of planning, among which we can mention:

- PDDL2.1 and following for temporal planning;
- PDDL/M for planning with external specialised modules.
- PDDL+ for planning with mixed discrete-continuous models.

PDDL/M and PDDL+ are not official languages adopted for use in any IPC, but were introduced in (Dornhege et al., 2009) and in (Fox and Long, 2006) respectively.

The language PDDL/M extendes PDDL to include *semantic attachments*, grounded predicates or changes of fluents evaluated by externally specified functions. It allows the specification of modules that verify conditions (*condition checkers*) or update the effects (*effect applicators*) when an action is invoked.

The language of PDDL+ is a variation of PDDL that support mixed discrete-continuous planning domains, using autonomous *processes* and *events*. Events are instantaneous effects initiated by the world in response to actions, while processes are changes that run over time and have continuous effects on numeric values.

3.2.2 Temporal Planning Languages

The commonly used language to express temporal planning problems is PDDL2.1 (Fox and Long, 2003) described in this section.

The language PDDL2.1

The temporal extension introduced in PDDL2.1 is made by introducing durative actions. An example of a durative action is shown in Figure 3.6. In particular, there are two forms of durative action: *discretised* and *continuous* actions. The difference depends on how numeric values change over the duration of the action, as the logical changes are considered to be instantaneous.

```
(:durative-action fly
    :parameters (?x - drone ?y - waypoint ?z - waypoint)
    :duration (= ?duration (timeToFly ?x ?y ?z))
    :condition (and
        (at start (inAir ?x ?y))
        (at start (accessible ?x ?z))
        (over all (launched ?x)))
:effect (and (at start (not (inAir ?x ?y))) (at end (inAir ?x ?z))))
```

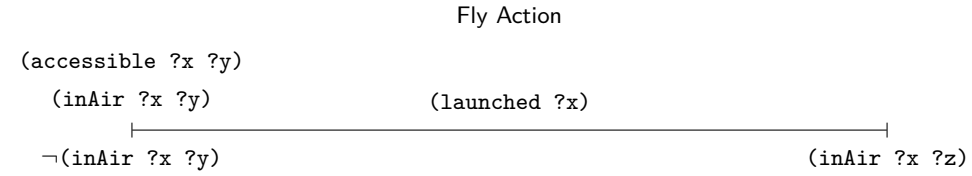


Figure 3.6: Example of a durative action in PDDL2.1 language for the task in Figure 3.1. The action is schematically drawn as a segment with length representing the duration. Preconditions appear above the segment, while effects are below.

In durative actions, the duration of the action must be indicated, whether it is fixed or variable, and the conditions are temporally annotated. The annotation of conditions makes explicit whether the associated proposition must hold at the start of the interval (the point at which the action is applied), the end (the point at which the final effects of the action are asserted) or over the whole interval between the start and the end. Numeric effects of discretised temporal actions and logical effects can be immediate (at the start of the interval) or delayed (at the end of the interval). For continuous durative actions some numeric variables can increase or decrease according to specified rates of change over time. The symbol `#t` is used to refer to the continuously changing time from the start of a durative action and throughout its execution.

This language allows actions to occur concurrently, giving rise to the problem of the validation of a plan. A necessary condition for a plan to be considered valid is that no logical condition can be both asserted and negated at the same instant and the rule of *no moving targets* is adopted: no two actions can simultaneously make use of a value if one of the two is accessing the value to update it. For numeric quantities, no numeric value can be accessed and updated simultaneously at the start or end point of a durative action, and, in case of continuous effects, during the entire duration of the action. Multiple simultaneous updates of numeric variables can be allowed only in the case of commutative operations. The *no moving target rule* has a consequence that mutually exclusive relations cannot hold at the same time, but they must be separated by a *non-zero* time separation ϵ. By contrast, if two end points of actions are not conflicting, they can be executed simultaneously. The value of ϵ is decided by the domain designer.

In addition, a metric to be minimised or maximised can be specified.

Further Evolution of PDDL2.1

From PDDL2.1 the language of the planning competition evolved adding further features:

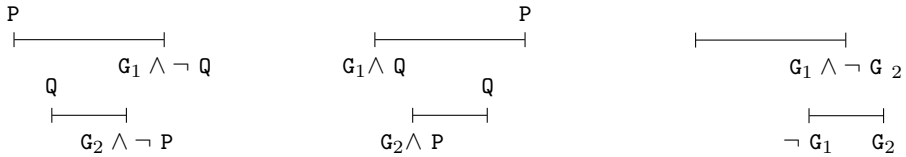


Figure 3.7: Three examples of domains with required concurrency taken from (Cushing et al., 2007). The initial state is {P,Q}, while the goal condition is $G_1 \wedge G_2$. The preconditions are indicated on the top, while effects are indicated at the bottom of the bar representing the action.

- PDDL2.2: the language of the 4^{th} planning competition, it introduces the timed initial literals (predictable exogenous events) and derived predicates (axioms) (Hoffmann and Edelkamp, 2005);
- PDDL3.0: used in the 5^{th} planning competition, added trajectory constraints and preferences or soft goals (Gerevini et al., 2009);
- PDDL3.1: used in the 6^{th} planning competition, added objective-fluents (SAS+ (Bäckström and Nebel, 1995)). With them, functions can have any object type and not only numerical ranges (Kovacs, 2011).

3.2.3 Temporal Expressiveness

When using a language in which actions have timed preconditions and effects, it is possible to distinguish between temporally simple or temporally expressive problems (Cushing et al., 2007). The difference between the two categories is in the presence of *required concurrency*. A planning problem has required concurrency when all the solutions are concurrent. Examples of problems with required concurrency are shown in Figure 3.7. In the case of temporally simple problems (which do not have required concurrency), it is possible to demonstrate that they can be reduced in linear time to classical planning problems where durations are ignored and every action is collapsed into a single transition. This is not possible for temporally expressive problems. The work in (Cushing et al., 2007) demonstrates that the PDDL2.1 language is a temporally expressive language capable of modelling problems with required concurrency and therefore problems modelled with this language cannot be generally solved as classical planning problems.

3.3 Two Approaches for Temporal Planning

Research in temporal planning follows two approaches to handle time. The first one is based on a decomposition of planning and scheduling, while the second considers the two parts in a unique framework.

3.3.1 Independent Planning and Scheduling

A first approach to solve problems with time and resources is to partition the problem into two independent parts: the planning and the scheduling parts. With this decomposition,

the planning problem is solved by first finding a set of actions to achieve a goal, looking only at the causal reasoning, while in the second step the actions are scheduled by some scheduling algorithm, usually minimising the makespan. This approach is used in several applications in the manufacturing and logistic sectors, where the planning phase is often performed by human experts. A recent example is offered in (Kameshwaran et al., 2013), in which planning and scheduling techniques are used in the iron mining industry.

Although this approach can be successfully applied to some problems, it is too simplistic when, for example, the temporal constraints are too tight, so that a plan does not have a feasible schedule.

3.3.2 Integration of Planning and Scheduling

An alternative approach to the decomposition of planning and scheduling is to consider the two parts coupled. A way to extend a classical planning problem to handle time and resources is to use time-stamped states, so that temporal constraints are considered during planning. Another way to embed time in planning is to not consider preconditions and effects of actions as explicitly distinguished, but to specify changes and persistence constraints over time. While this latter approach has a timed-oriented view, that led to several planners, such as IxTeT (Laborie and Ghallab, 1995) and the more recent EUROPA (Frank and Jónsson, 2003), the former has a state-oriented view similar to the classical planning approach. Thus, the techniques that are used in classical planning can be extended to handle time.

3.4 Planning as a Search Problem

A planning problem can be seen as a search problem. There are two different approaches, based on the search space adopted:

- state space planning: the search space is given directly by the possible states;
- plan space planning: the search space is the set of partially specified plans and the search transitions are "refinement operations".

In the formulation of planning as state space search, nodes are states of the planning problem $\mathcal{P}$, while the edges are the possible actions in $\mathcal{A}$. Two approaches can be adopted during the search: the first, called *forward search*, starts from the initial state and applicable actions are added until a goal state is reached; the second approach, called *backward search* or *regression*, starts from a goal state and explores states obtained applying the inverse of an action until the initial state is reached.

Partial order planning generally refers to the formulation of planning as a search problem where the states of the search are not the states of the world, but partially specified plans. In this formulation, the initial state is usually an empty plan, and the state transitions are refinement operations that add constraints to complete a partial plan. In particular, there are two types of constraints: *ordering constraints* and *binding constraints*, where the first refers to the order of actions in the plan, and the second is a constraint of the form $x = y$ or $x \neq y$. A solution is achieved when all the ordering and binding constraints are consistent, and every sequence of totally ordered and totally instantiated actions that

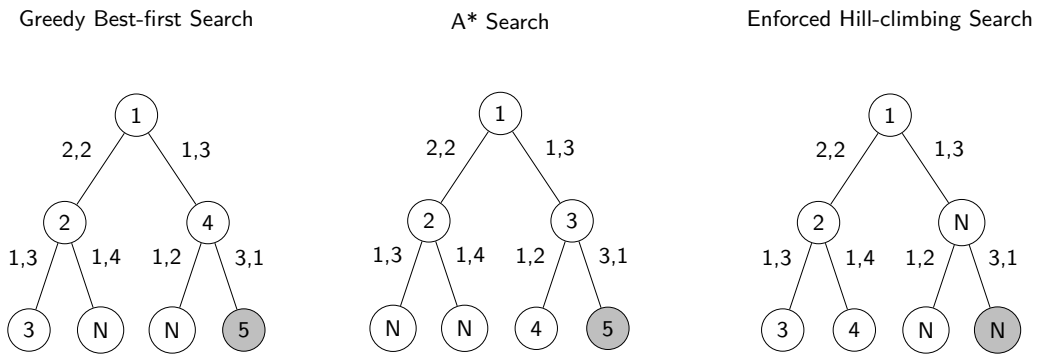


Figure 3.8: Informed search algorithms. The labels in the nodes represent the order of expansion, while label of the edges represent the cost.

satisfies the constraints, applied to the initial state of the state transition system, produces a state that contains all the goals. Plan space search usually exploits the *least commitment principle*, which states that a constraint is not added to the plan unless it is strictly needed.

The main focus of this thesis is forward state-space search planning, which is the framework used in POPF2. This planner also exploits some partial-ordering of actions during plan search. In the rest of this section we include an overview of the main techniques developed using this framework, including the basic search algorithms and the commonly used heuristics.

3.4.1 Search Algorithms

Search algorithms are divided into two categories:

- uninformed search (or blind search): the expansion of the explored states does not exploit any information, other than the problem specification. The most widely considered algorithms are *breadth-first search*, *depth-first search* and *uniform cost search*.
- informed search: this class of algorithms uses extra information to favour some nodes over others. These algorithms exploit a *heuristic*, an estimation of the cost to reach a goal state, therefore they are also known as heuristic search algorithms. Examples of such algorithms are *greedy best first search* (Pearl, 1984), $A*$ (Hart et al., 1968) and its variations, and *enforced hill climbing* (Hoffmann and Nebel, 2001).

3.4.2 Informed Search

The heuristic value $h(s)$ of a state s can be defined as the estimated cost of the cheapest path from s to a goal state (Russell and Norvig, 2003). This value is used in the evaluation function $f(s)$, which is an estimation of the cost of finding a solution, therefore it is used to determine the priority in which states are expanded during the search: the states with the lowest $f(s)$ are the ones expanded first. Varying the definition of the evaluation function we obtain different algorithms. The family of global search algorithms that uses the heuristic in the evaluation function is called *best first search*. Heuristics can also be exploited in local

search algorithms such as Enforced Hill Climbing. Figure 3.8 shows the search trees explored by some of the informed search algorithm described in the following.

Greedy Best First Search. This algorithm belongs to the *best first search* family, where the evaluation function of a state s is simply the heuristic value:

$$f(s) = h(s) \tag{3.1}$$

It does not take into account the cost or the length of the solution, but the idea behind it is to find a solution as quickly as possible.

A* Search. This is a *best first search* algorithm, using the evaluation function

$$f(s) = g(s) + h(s), \tag{3.2}$$

where $g(s)$ is the cost to achieve the state s. Provided that the heuristic function is admissible (i.e. it does not overestimate the real cost of achieving the goal state), this algorithm finds optimal solutions.

Weighted A* Search. This algorithm (Pohl, 1970) is a variation of A*, where the evaluation function is composed of the sum of the cost to achieve a state and a weighted cost of achieving a goal from this state.

$$f(s) = g(s) + w \cdot h(s), \tag{3.3}$$

with weight $w > 1$, representing the trade off between optimality and speed in solving. It should be noted that uniform cost search can be seen as a weighted A* search with $w = 0$.

Enforced Hill Climbing (EHC). This is a local search algorithm. It was first introduced in (Hoffmann and Nebel, 2001) and it is a variation of the hill climbing algorithm. Here, the closest better successor of a state s is expanded. The closest better successor is found with a breadth first search from s. When the breadth first search finds a better state s' for which $h(s') < h(s)$ the queue is deleted, and the breadth first search restarts from s'.

3.5 Heuristic Evaluation

One of the erliest formulation of planning as heuristic search was in the work of (Bonet and Geffner, 2001), where a novel domain-independent heuristic was formulated. This was based on *relaxed* problems: simpler problems obtained from the original by relaxing constraints. From the planning problem $\mathcal{P}$ the relaxed planning problem $\mathcal{P}'$ is generated. From any state s, $h'(s)$ is defined as the optimal cost for solving the relaxed problem $\mathcal{P}'$ and it is also a lower bound on the optimal cost for solving the original problem $\mathcal{P}$. The value $h'(s)$ can be used as an admissible heuristic function. Further relaxation or different approaches have been introduced, producing various heuristics. Five major approaches are explored in the literature:

- Delete relaxation: the problem is simplified by neglecting the delete effects of actions. One of the most influential variants is FF (Hoffmann and Nebel, 2001).

- Abstraction (Edelkamp, 2001; Haslum et al., 2007): the original problem is made simpler by considering a smaller space, not considering all the states as distinct. The abstract state space is derived from the original state space by an abstraction function α, that defines which states should be distinguished. The heuristic value of a state s is then derived from the cost of the cheapest path from $\alpha(s)$ to a goal state in S^{α} space. Depending on the abstraction function different heuristics can be derived.

- Critical paths (Geffner and Haslum, 2000): the heuristic is calculated as the critical path length of a concurrent solution for a simplified problem.

- Landmarks (Hoffmann et al., 2004; Richter et al., 2008): landmarks are defined as conjunctions of actions that all the plans must include. Landmarks can be used to derive a heuristic value or used in conjunction with a delete relaxation.

- Network flow (van den Briel et al., 2007): the heuristic is calculated by balancing the number of times each fact is produced and the number of times it is consumed.

3.5.1 Classical Planning Heuristics

In this section we first describe some of the classical planning heuristics, whose extensions to handle temporal aspects and numeric resources are presented in the remainder of the section.

The Additive Heuristic h^{add} and the Max Heuristic h^{max}

The additive h^{add} and the max h^{max} heuristics (Bonet and Geffner, 2001) are calculated from a delete relaxed problem S'. The additive heuristic is based on the assumption of the independence of the goals of the relaxed problem, meaning that the actions are assumed to not negatively interact with each other. In this way the h^{add} is calculated recursively by adding the costs of achieving the goals, G:

$$h^{add}(s) = \sum_{p \in G} g_s(p). \tag{3.4}$$

where $g_s(f)$ is the estimated cost of achieving a fact f from a state s, and it can be calculated as:

$$g_s(f) := \begin{cases} 0 & \text{if } f \in s \\ \min_{a \in \mathcal{O}(f)} \left[cost(a) + g_s(pre(a)) \right] & \text{otherwise} \end{cases} \tag{3.5}$$

where $\mathcal{O}(f)$ indicates the set of actions of the problem that add the fact f. Although h^{add} can be computed in polynomial time, it is not admissible.

Similarly to h^{add}, the max heuristic h^{max} is calculated using the cost of achieving a goal, but here the costs of the goals are not summed, but only the maximum value is considered:

$$h^{max} = \max_{p \in G} g_s(p) \tag{3.6}$$

Unlike the additive heuristic, the max heuristic is admissible, although it is often less informative.

The Graphplan Heuristic h^G

The Planning Graph was first introduced by Blum and Furst in (1997), as a structure to guide the search for plans. In particular, a Planning Graph is a directed and layered graph in which two kinds of layers are alternated: the *proposition layer*, containing the *proposition nodes*, and the *action layer*, containing *action nodes*. Starting from the first proposition layer, the nodes are represented by the propositions in the initial state, and the first action layer contains all the applicable actions from the initial state. The subsequent layers are built adding all the propositions that can possibly be made true and all actions that are possibly applicable given those propositions. The edges of the Planning Graph are the relations between actions and propositions, thus there are three kind of nodes: *precondition-edges*, connecting propositions at layer i to actions at layer i, *add-edges*, and *delete-edges*, connecting actions at layer i to propositions at layer $i+1$. A crucial part of the Planning Graph is the detection of *mutually exclusive* relations (*mutex*). In particular, two propositions p and q in the layer i are mutex if every action in the preceding action layer that has p as a positive effect is mutex with every action that has q as a positive effect and there is no single action in the layer $i-1$ that has both p and q as positive effects. Two actions are mutex if they are dependent (one action deletes a precondition or a positive effect of the other) or have mutex preconditions. The identification of these mutex relations helps the Graphplan algorithm to find the plan, searching backwards from the goal to the initial state.

As noted in (Bonet and Geffner, 2000) the Graphplan algorithm can be understood as a heuristic search algorithm, where the heuristic value $h^G(s)$ is given by the index i of the first layer in the graph that contains the propositions in s without any mutex relations.

The FF Heuristic h^{FF}

The FF heuristic (Hoffmann and Nebel, 2001) is derived using the Planning Graph structure for relaxed planning problems, where the relaxation comes from ignoring the delete-lists. The resulting *Relaxed Planning Graph* (RPG) contains no mutex relations and a solution can be found without the need for backtracking. An example of a Relaxed Plan is reported in Figure 3.9.

The FF heuristic is then defined as the length of the relaxed plan, computed from the RPG. The relaxed plan is extracted, selecting iteratively the best supports of the goals and of the preconditions of the actions already selected. The best support action is the cheapest achieving action with regard to the cost estimates (i.e. the index of the layer in which it first occurs in the graph). This heuristic is not admissible because the solution is not necessarily optimal; the actions are not chosen to minimise the size of the complete set of supporting actions, but only to minimise the cost of the support action for each goal or precondition as it is considered. To find an optimal relaxed plan is an NP-hard problem, but the greedy approximation is generally very accurate.

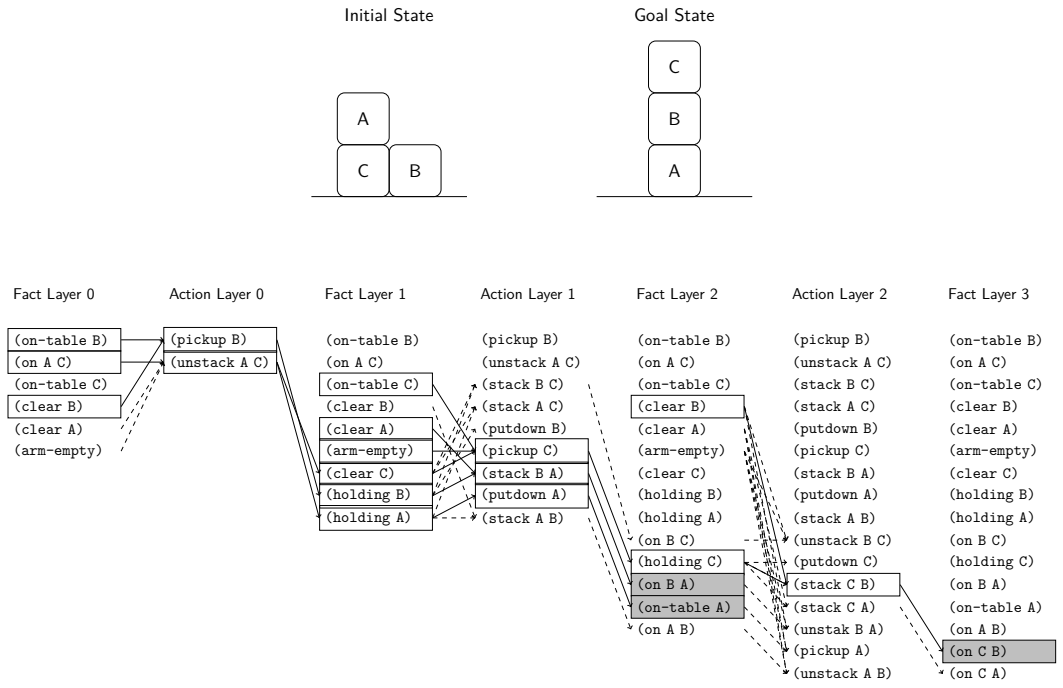


Figure 3.9: Example of Relaxed Planning Graph for the Blocksworld domain. The dashed lines represent the preconditions and the effects of actions, solid rectangles are the goal facts, and the empty rectangles are the actions and the preconditions to achieve the goals.

Critical Path Heuristic h^m

The critical path heuristic was introduced in (Geffner and Haslum, 2000) and it can be defined as the cost of achieving a set of atoms A from a state s by achieving the most costly subset of size $m' \leq m$ in A.

$$h^m(A) := \begin{cases} 0 & \text{if } A \subseteq s, \\ \min_{a \in \mathcal{O}(f)} \left[cost(a) + h^m(pre(a)) \right] & |A| \leq m \text{ and } A \nsubseteq s, \\ \max_{B \subset A, |B|=m} h^m(B) & \text{otherwise} \end{cases} \tag{3.7}$$

This set of heuristics h^m, for $m = 1, 2, ...$ are all admissible, and they are polynomial in both the number of actions and the number of atoms in the problem. The higher the value of m, the more expensive the heuristic is to compute, although the heuristics are increasingly informative. The case of $m = 1$ is proven to be equivalent to the max heuristic $h^1 = h^{max}$, while for $m = 2$ we have $h^2 = h^G$.

3.5.2 Numeric Heuristics

In the following, two heuristics which extend h^{FF} and take into account metric values are described.

The Metric FF Heuristic h^{mFF}

A heuristic that takes into account numeric resources (but not temporal aspects) is the metric-FF heuristic (Hoffmann, 2003), an extension of h^{FF}. In this work, a state is defined by the set of propositions P that are true and a set of numeric variables V in the domain of the rational numbers $\mathbb{Q}$. The numeric variables v_i can be combined together in arithmetic expressions *exp*, which can be subject to constraints of the form (*exp comp exp*$'$), where $comp \in \{<, \leq, =, \geq, >\}$, and numeric effects are in the form (v_i *op exp*), where $op \in \{:=, +=, -=, *=, /=\}$. In particular only linear tasks are considered, where the numeric variables are used in linear expressions. These expressions can be reduced to a *linear normal form* (LNF), where all the expressions can be written as a sum of variables with strict positive weights. Thus, all the constraints can be reduced to the form:

$$\left(\sum_{j \in X} c_j * v_j + c\right) \geq [>]\, 0, \text{ where } c_j, c \text{ are constants in } \mathbb{Q} \tag{3.8}$$

In the relaxed planning graph the numeric expressions are relaxed by ignoring the decreasing effects, so that when a condition is achieved in the RPG, it remains satisfied in the subsequent layers.

The LPRPG Heuristic h^{lprpg}

This heuristic (Coles et al., 2008) is designed to extend the h^{mFF} heuristic to improve performance in the presence of numeric effects. The heuristic is based on a combination of a linear programming (LP) problem for the numeric part and an RPG for the propositional part of the planning problem. The LP problem is built by relaxing a map of the RPG to a Mixed Integer Program (MIP). The MIP contains the variables $v_{0,l}, ..., v_{n,l}$ for each fact layer l, corresponding to the numeric variables of the problem. Then, for each action layer l the variables $a_{0,l}, ..., a_{m,l}$ represent actions that have been applied or not. The $v_{i,l}$ variables are constrained restricting the value of the variables $v_{i,l+1}$, according to the behaviour of the actions that influence such variables. The LP problem is used to calculate the bounds that each variable can assume in each layer of the RPG and to determine the actions applicability in the RPG construction. The LP is also used during the extraction of the relaxed plan to achieve the numeric preconditions of actions.

3.5.3 Temporal Heuristics

The extensions of the heuristics to handle time are presented in the following.

The Temporal Critical Path Heuristic h^m_T

The work (Haslum and Geffner, 2001) extends the critical path heuristic for planning problems with time and resources. In the class of temporal problems analysed in their work, the actions are assumed to have durations, all the preconditions are assumed to be satisfied at the starts of actions and the effects happen at their ends. This is only a subset of all the possible durative actions that can be modelled in PDDL. In this work, and in general

when considering temporal planning problems, the cost of a plan is not given by the number of actions needed to achieve a goal state, but it is given by the least time t such that a goal state is achieved. Thus, the heuristic h_T^m is an extension of the h^m heuristic where the cost of applying an action is not unitary but it is given by its duration.

$$h_T^m(A) := \begin{cases} 0 & \text{if } A \subseteq s, \\ \min_{a \in \mathcal{O}(f)} \left[dur(a) + h_T^m(pre(a)) \right] & |A| \leq m \text{ and } A \not\subseteq s, \\ \max_{B \subset A, |B|=m} h_T^m(B) & \text{otherwise} \end{cases} \tag{3.9}$$

This heuristic preserves its admissibility in the presence of renewable and consumable resources, although its informativeness is compromised.

The TRPG Heuristic h^{TRPG}

Temporal Relaxed Planning Graphs are extensions of the RPG that can cope with time. A first extension was proposed in (Smith and Weld, 1999), where the language adopted does not have the capability of modelling required concurrency. An alternative extension is provided in (Coles et al., 2009b), where an action a is divided into two snap actions $a_{\vdash}$ and $a_{\dashv}$, respectively the start point and the end point of the action. The RPG is augmented by assigning a time-stamp to each fact layer . To capture the duration, the $a_{\dashv}$ is assigned to an action layer labelled with the earliest time in which the snap action can appear, while the actions that are already executing (but not yet finished) appear in the TRPG at action layer $t = 0$. The heuristic value is associated with the length of the plan extracted from the TRPG.

3.5.4 Search Enhancement

Many satisficing planning systems exploit various enhancement techniques derived from the use of heuristic evaluation.

Preferred Operators

This term was officially introduced in (Helmert, 2006), indicating that a subset of applicable operators of a state s contains operators considered promising under some *criteria*. The first example of preferred operators was actually before this definition, in the work of (Hoffmann and Nebel, 2001), where the *helpful actions* are a side effect of the computation of h^{FF}. Helpful actions are identified as the actions that belong to the relaxed plan and are applicable in the current state. The planner FF makes use of the helpful actions during the local search (EHC), where they are expanded first.

3.6 Temporal Numeric Planners

In this section we present different temporal numeric planners and we explore their different capabilities. Amongst the most recent planners we consider:

- LPG-td (Gerevini et al., 2006): this planner is based on a stochastic local search procedure (Walkplan) to solve planning graphs, using as heuristic evaluation a

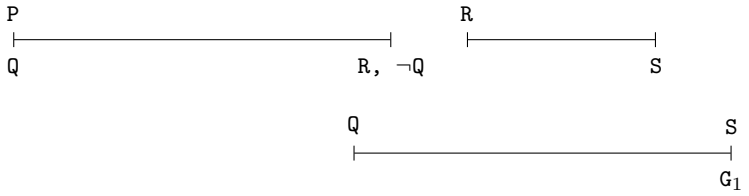


Figure 3.10: Example of required concurrency not handled by SAPA and TFD.

weighted function that estimates the search cost and the temporal cost. The search cost of adding an action a at a level l is estimated by a relaxed temporal plan.

- SGPlan (Chen et al., 2006): is based on partitioning a planning problem into sub-problems with global constraints. Each sub-problem is then resolved by metric-FF-based planner, and the solutions are combined resolving global constraints.

- Sapa/TILSapa (Do and Kambhampati, 2003; Kavuluri and U, 2004): Sapa and TILSapa (the extension to handle Timed Initial Literals) are planners that use forwards chaining A* search, where the search is performed in the space of time-stamped states. The transition function is given by the insertion of a durative action or the advancement of the time by a certain increment. Sapa does not consider any time-point between the start and the end of an action, providing only a partial management of the required concurrency. An example of concurrency not handled by SAPA is shown in Figure 3.10.

- TFD (Eyerich et al., 2009): this planner is a temporal planner based on Fast Downward (Helmert, 2006). As in SAPA, the search space is the set of time-stamped states. The planner performs an A* search where the evaluation function is given by the sum of the time-stamp of the state and the heuristic value. The heuristic evaluation is based on a variation of the inadmissible context-enhanced additive heuristic used in FD.

- MIPS/MIPS-XXL (Edelkamp, 2003; Edelkamp and Jabbar, 2008): the Model Checking Integrated Planning System (MIPS) and its extension MIPS-XXL are temporal planners that follow the approach *plan first schedule later*. The durative actions are first translated to instantanous ones and a sequential plan is found with an weighted A* algorithm, where the evaluation function of a state s is $f(s) = cost(s) + 2 * h(s)$. The heuristic evaluation is based on the relaxed plan graph or pattern database heuristics. The scheduling of actions is executed with a critical path analysis algorithm.

- POPF2 (Coles et al., 2009b): it is based on a forward-chaining state-based search strategy that can support partial-order planning in the solution of temporal-numeric problems. More details on this planner are given in the next section.

- CPT (Vidal and Geffner, 2006; Vidal, 2011): this is an optimal temporal planner, based on a formulation of planning as a set of constraint satisfaction problems. In a preprocessing step, the planner computes lower bounds on the time of application of each action a and each pair of atoms $\{p, q\}$ using the heuristic value $h^2_T(a)$, and it

	Temporal	Numeric	Required Concurrency	TIL	Duration Inequalities
LPG-td	✓	✓		✓	
SGPlan	✓	✓		✓	
Sapa	✓	✓	✓(partial)	✓	
TFD	✓	✓	✓(partial)		
MIPS	✓	✓		✓	
POPF2	✓	✓	✓	✓	✓
CPT	✓		✓		
UPMurphi	✓	✓	✓	✓	✓

Table 3.1: Summary of the capabilities of the most recent temporal numeric planners

computes the time of advancement of actions with h^1_T. After this step the variables are instantiated and the constraints are built. The planner then branches different states by selecting and fixing flaws of the state, according to the partial ordering paradigma and the constraints are propagate to prune the search space. CPT relies on a discretisation of time at the smallest common factor of the durations of actions in the domain. This makes it infeasible for problems with widely differing action durations.

- UPMurphi (Della Penna et al., 2009): this planner handles continuous processes using the discretise and validate approach. The planner iteratively finds a plan with a finer time granulation when the validator finds an invalid plan.The search algorithm is A*, considering only the cost to achieve a state and no guidance on the cost to achieve the goal. The lack of heuristic limits the size of the problems that the planner can solve.

The capabilities of each planner are summarised in Table 3.1. The critical feature is the *required concurrency*. Many of the temporal planners do not actually handle concurrency correctly. LPG-td, SGPlan, and MIPS consider only sequential solutions and reschedule these using temporal information. Sapa and TFD handle a subset of the required concurrencies, while only POPF2, UPMurphi, and CPT are capable of solving these kind of problems.

3.6.1 Other Temporal Numeric Planning Approaches

Within the context of numeric temporal planning problems, an alternative approach is the *Domain Predictive Control* (DPC) approach of Löhr *et al.* (2012). This is a planning based method for modelling and solving hybrid dynamic systems. Physical processes are expressed as systems of differential equations, solutions of which are pre-computed and stored for look-up during the planning process. In DPC, a set of discrete and pre-defined input signals is treated.

Löhr *et al.* (2014) generalises DPC (*symbolic DPC*) to deal with continuous bounded input signals, expanded as numeric parameters of actions and symbolic goals. The planning problem consists of achieving the symbolic goals that guarantee the reachability of numeric goals. The actual values of the numeric variables are found in a second step by using a function generating successor numerical states from the previous ones over fixed temporal horizons.

Although these approaches are successful in dealing with metric goals in hybrid dynamic systems, the temporal complexity of plans is limited. Actions of the planning problem are

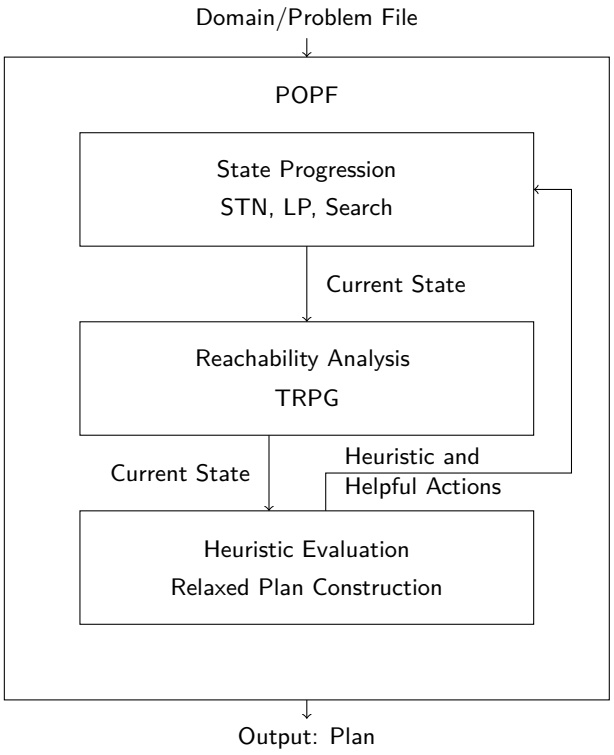


Figure 3.11: POPF2 architecture.

only sequential and concurrencies of the physical system are captured by instantiating new actions with combined effects. This limits the number of actual operations that can be modelled.

3.7 Temporal Planning in the POPF2 Framework

The planner POPF2 (Coles et al., 2010) (Figure 3.11) is a forward search temporal planner that avoids early commitment to the ordering of actions encoding plans as a partial orders.

The temporal reasoning part of the planner exploits the mechanisms introduced in its predecessors CRIKEY3 (Coles et al., 2009b) and COLIN (Coles et al., 2012), capable of handling durative actions with discrete and continuous effects, respectively.

3.7.1 CRIKEY3

In CRIKEY3 the durative actions are split into start and end actions, referred to as *snap actions*. Each state S in the search space is composed of F, the set of propositional facts and fluent values, E, the ordered list of events already started but not yet finished, and T, a collection of temporal constraints over the actions in the plan to reach F. This extension allows the recording of invariants of any actions that have started, but not yet finished. In that way, when the planner considers which actions have satisfied preconditions, actions that

violate any of the these active invariants are excluded. CRIKEY3 also identifies *compression safe* actions (Coles et al., 2009a), which are actions that can be considered as a single decision point without compromising soundness or completeness of the search. This allows a more refined temporal reasoning when a plan involves concurrent activities.

Temporal constraints are captured by a Simple Temporal Network (STN), built at each state in the search space. The heuristic is evaluated by an extension of the RPG used in metric-FF, called the Temporal Relaxed Planning Graph (TRPG). In the TRPG action layers contain both start and end actions, while the fact layers contain their positive effects. The end actions appear at the time delayed after theirs starts by the minimum duration. The state representation in CRIKEY3 is therefore given by the following definition.

Definition 7 (State in CRIKEY3) *A* state *in CRIKEY3 is a 5-tuple:*

$$s := \langle F, V, Q, P, C \rangle,$$

where F is the set of atomic propositions that hold in the state, V is a vector of values of the numeric variables, Q is the list of actions that are started, but not yet finished, P is the plan to reach the current state, and C is a list of temporal constraints over the steps in P.

Timed Initial Literals (TILs) are handled by creating dummy *TIL actions*, comprising the effects of the TILs at thier given time-points. A TIL action can be added in P only if all earlier TILs actions have already been added and if they do not delete the invariant of any open actions.

3.7.2 COLIN

The planner COLIN can handle linear continuous change. Continuous changes imply that the temporal interactions cannot be separated from reasoning about numeric variables. The state representation in COLIN extends the tuple in Definition 7 adding the vector V' which, for each step i of the plan, records the values of numeric variables after the application of an action. Numeric preconditions and effects of actions are represented as constraints over the values $v_i \in V(i)$ and $v'_i \in V'(i)$. In this planner, the plan is formulated as a Linear Program (LP) containing both the temporal constraints and the numeric constraints imposed by the actions. The LP is solved at every node in the search space, so its solution must be as much efficient as possible.

3.7.3 POPF2

The partial ordering in POPF2 is achieved by delaying the commitment to ordering decisions, time-stamps and the values of numeric parameters. In order to deal with the partial ordering, in POPF2 a search state s is defined as follows:

Definition 8 (State in POPF2) *A state is a tuple:*

$$s := \langle F, V, Q, P, C, V', F^+, F^-, FP, V^{eff}, V^{cts}, VP, VI \rangle,$$

where F, V, Q, P, C and V' are the same as in CRIKEY3 and COLIN, and F^+ (F^-) is a map between the facts and the indices of the actions by which a fact has been most recently

added (or deleted), FP is the map between the facts and pairs $\langle i, d\rangle$ indicating that a fact p at step i is required to hold during an open or half closed interval. For all the numeric variables $v \in V$, $V^{eff}(v)$ contains the index of the most recent step having an instantaneous effect on v, $V^{cts}(v)$ records the information about the variables subject to continuous changes. The set VP contains, for each numeric variable v, the set of indices where $i \in VP(v)$ depends on the value of v. The set VI contains the pairs of indices (i, j) indicating respectively the step at which each action begins and ends.

When a state is updated adding an action, the ordering constraints contained in C are checked for consistency. In the absence of continuous numeric effects the ordering constraints are checked by an STN, while in the presence of continuous numeric effects the planner uses a similar adaptation of the temporal-numeric consistency approach of COLIN, but due to the partial ordering the value of the variables v are no longer linked to the last step of the plan, instead each variable has a new time-step representing the current time-step and it is constrained with respect to this time-step.

Heuristic

The heuristic evaluation used in POPF2 is based on the temporal relaxed planning graph (TRPG). The relaxation of numeric state variables used in POPF2 is based on the same principle introduced in metric-FF (Hoffmann, 2003) of ignoring negative effects of linear numeric functions, as explained in Section 3.5.2. This means that, for the heuristic evaluation, the numeric effects are written in a *Linear Normal Form* as in Equation 3.8. In this way, it is possible to consider the reachable values of each variable as an interval between the minimum and maximum values that the variable can assume when modified by actions. The reason why effects must be expressible in LNF is because the relaxation must converge. To avoid building infinitely many layers, asymptotical behaviours such as

$$v \mapsto v \cdot \alpha, \tag{3.10}$$

with $\alpha < 1$, are excluded. In contrast with metric-FF, when there is an effect of the type

$$v \mapsto a \cdot v + \sum w_i \cdot b_i + c, \tag{3.11}$$

POPF2 uses infinity handling, as proposed, but not implemented by Hoffmann (Hoffmann, 2003). Infinity handling allows increasing effects on metric fluents to be applied arbitrarily often once the effect appears in the reachability analysis. In plan extraction, the number of applications of the effect required to satisfy the goal is computed by considering the necessary increase over the current value of the fluent in order to reach the goal threshold. To see why infinity analysis is necessary in POPF2, we consider the case in which a durative action has an increase effect on a metric fluent at its end point. Without infinity analysis, once the action can be applied in the TRPG, the end point would have to be applied at multiple time-points (to obtain the accumulated effects of multiple applications of the increase effect) and these would be separated only by ϵ since the relaxation would allow the same action end point to be applied repeatedly once its preconditions were satisfied. As a result, if the duration of the relaxed plan in the TRPG is d, the TRPG would require approximately d/ϵ

layers, which will usually be many thousands, making the TRPG impractical to construct.

The TRPG is modified to handle continuous linear changes and the partial-order. Since the current state no longer has fixed values for variables subject to continuous change, the bounds for the variables are used. Moreover, it is assumed that the continuous changes are available as soon as the action has started and if an action that consumes a resource has already started, it consumes as much as possible of the resource before the start of the current TRPG construction.

To handle the partial-order, all facts in the current state are added to a fact layer with the earliest times at which each fact could be achieved (this is used to improve the heuristic estimates when the estimated makespan is also used). The earliest time a fact p is available is the maximum of the minimum time in which the fact p is added and the minimum time at which the fact p can be deleted plus an epsilon time (because either it can be achieved by its last achiever or it must be reachieved after its last deleter).

3.7.4 Search Algorithms

The search algorithms used in POPF2, as well as by its predecessors CRIKEY3 and COLIN are:

- Enforced Hill Climbing with Helpful Action pruning, where here helpful actions are defined as all actions in the relaxed plan whose preconditions are satisfied in the current state. It should be noted that, due to the partial ordering, the current state is not necessarily the first layer of the relaxed plan.

- Best First Search. If EHC fails to find a solution, the Best First Search is resorted from the initial state. The information on helpful actions is exploited when expanding a node: the first children evaluated are the ones obtained applying helpful actions.

Chapter 4

Bounded Trajectory Management Problems

In this chapter we start introducing numeric exogenous events, which we model as an extension of timed initial literals in PDDL. Introducing such features gives rise to a new class of problems, that we call *Bounded Trajectory Management Problems*, which we formally define. The methods to solve this class of problems are described in the second part of the chapter: we introduce the *lookahead* heuristic and two alternative search strategies that we call *Mixed Search* strategies.

4.1 Introducing Timed Initial Fluents

Timed Initial Literals (TILs) were first introduced in PDDL during IPC-4 (Hoffmann and Edelkamp, 2005) to model deterministic unconditional exogenous events that become true or false at certain time-points. They are events known to the planner in advance; in fact they are specified in the initial state. Timed Initial Fluents (TIFs) are a natural extension of TILs that express changes in numeric fluents, allowing the addition or the deletion of resources independently of the actions in the plan.

The syntactic definition of TIFs follows the one of TILs:

```
<init> ::= (:init <init-el>*)
<init-el> ::=numeric-timed-initial-fluent(at <number> (= <f-head> <number>))
```

An illustrative example of TIFs, given a fluent `(x)` defined in the domain file, is:

```
(at 1 (= (x) 1))
(at 2 (= (x) 4))
```

where the fluent `x` assumes a value of 1 at time 1 and a value of 4 at time 2. The semantics of PDDL including TIFs is a straightforward extension of the one of PDDL2.2.

With this syntax we only consider exogenous events that assign values to fluents and do not allow relative changes by increase or decrease effects.

In real world problems, TIFs are often used to describe discretised functions of time that are independent of the actions chosen by the planner. In particular, there is a class

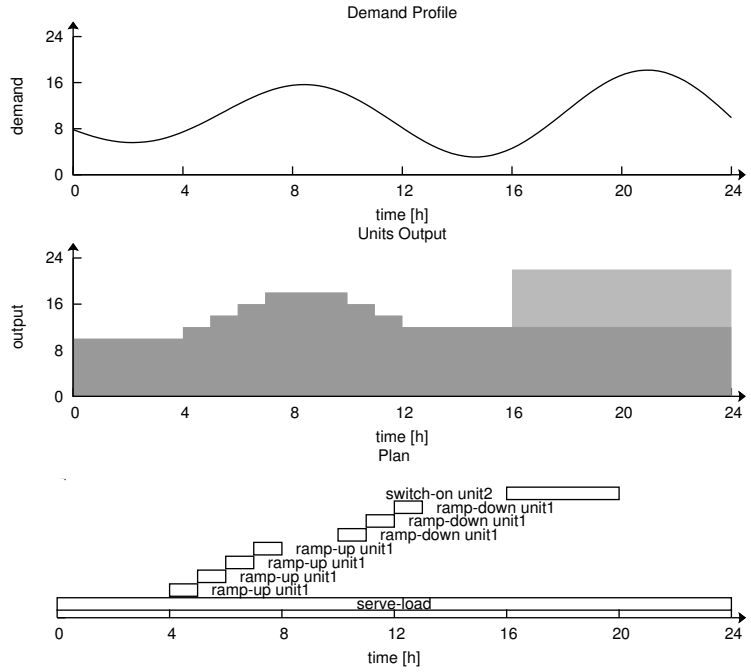


Figure 4.1: Example of UCP problem. On the top plot there is the predictable demand, while on the bottom part there is the generation output produced by the plan shown in the lower part of the figure.

```
(:action apply-til
   :parameters (?t0 ?t1 - dt)
   :precondition (and (is-now ?t0) (now ?t0) (is-before ?t0 ?t1)
      (is-now ?t1))
   :effect (and (now ?t1) (not (now ?t0))
      (assign (x) (x-at-time ?t1))))
```

Figure 4.2: The `Apply-til` action enforces at a given time-point the update of the numeric fluents modified by TIFs.

of real world problems, such as the Unit Commitment Problem (UCP), characterised by the presence of background functions (in this case the electrical demand profiles), against which the planner needs to actively react to maintain some numeric constraints through an extended period of time (Figure 4.1).

Another example where TIFs are useful to model some background function is presented in (Parkinson et al., 2014), where planning is used to reduce the downtime and the uncertainties on measurements of machine tools calibration. In this domain the calculated uncertainties depend also on the room temperature, which is independent of the actions. This temperature profile is provided as values every 0.01 hours, and it can be easily modelled with TIFs. Since they are not widely recognised by the planning community, the authors chose to express the behaviour with piece-linear functions that are applied subsequently.

```
(= (x) 2)
(at 1 (= (x) 0))
(at 2 (= (x) 10))
```

```
(now dt0)
(before dt0 dt1)
(before dt1 dt2)
(at 1 (is-now dt1))
(at 1.001 (not (is-now dt0)))
(at 2 (is-now dt2))
(at 2.001 (not (is-now dt1)))
(= (x-at-time dt0) 2)
(= (x-at-time dt1) 0)
(= (x-at-time dt2) 10)
```

Figure 4.3: Example of translation from TIFs to TILs, showing the original TIFs on the left and the translation on the right.

4.1.1 Compilation of TIFs into TILs

Although TIFs present a simple way to express numeric background events, it is possible to compile TIFs into a series of TILs. In the following we describe a way to compile them.

First a new type of object `dt` is created to represent a time-point at which a TIF occurs. For each TIF in the problem file, TIF_i, an object $\texttt{dt}_i$ is created.

For each fluent that is changed by a TIF, $f_{TIF}(\underline{x})$, (where $\underline{x}$ represents the PDDL object arguments of the PDDL function encoding the fluent) a copy function is created with the same arguments and an additional argument of the type `dt` ($f^*_{TIF}(\underline{x}, \texttt{dt}_i)$). These are initialised to their respective values in the initial state.

An action (shown in Figure 4.2) is added to the domain to enforce the updating of the value of the $f_{TIF}(\underline{x})$ functions. This action takes as parameters the two consecutive objects $\texttt{dt}_i$ and $\texttt{dt}_j$ that are added and negated by TILs in the problem file at the same time as the corresponding TIF. It also updates the proposition `(now ?dt)` that keeps track of the TIF time. This action requires the propositions `(is-now ?dt)` for the two values of `?dt`, $\texttt{dt}_i$ and $\texttt{dt}_j$, to be simultaneously true at the point at which it is executed. This is achieved using TILs that set the windows over which these conditions hold, as illustrated in Figure 4.3. Using this compilation, the *no moving targets rule* (Fox and Long, 2003) is applied: an action cannot simultaneously access to a variables when another one is updating it. This means that two actions using the same variable must be separated by a non-zero time gap. This restriction is imposed in order to favour conservative restrictions on compatibility of actions rather than have a more permissive requirement that is more difficult to check. In this specific case the *no moving target rule* enforces that the planner cannot apply an action that relies on $\texttt{(now dt}_i\texttt{)}$ being true at the same time as $\texttt{(apply-til dt}_i\ \texttt{dt}_j\texttt{)}$ is applied. The proposition $\texttt{(now dt}_k\texttt{)}$, where $\texttt{dt}_k$ is the last TIF (in the example `dt2`) is added to the goal to ensure that all the `apply-til` actions are applied subsequently, updating the functions $f_{TIF}(\underline{x})$ all the way to the end of the plan.

In the resulting plan, the series of `apply-til` actions will appear alongside the proper actions, managing the behaviour of the TIFs by the compiled process. This compilation is correct, but adds a burden to the planner, which must reconstruct the behaviour of the TIFs enforcing the application of the `apply-til` actions alongside the construction of its plan. This can add complications to the search space (depending on the strategy the planner uses) that can prevent the planner from finding a plan.

4.1.2 Difference between TILs and TIFs

TIFs might appear as a straightforward extension of TILs, because they both change state variables in a way that is independent of the actions of the planner, however they introduce a new degree of complexity.

The major difference between TILs and TIFs lies in their interaction with invariant constraints, established as `over all` conditions within concurrent actions or other trajectory constraints such as `always` conditions. In the case of a propositional domain, an active constraint involves only a proposition that is required to be *false* or *true* throughout the duration of the action. If a TIL modifies that proposition, it makes the precondition of that action *true* (or *false*) and we can deduce that the action cannot intersect the time-point of the TIL.

In the case of numeric fluents, according to PDDL2.1 syntax, the condition has the form:

```
<f-comp> ::  = <binary-comp> <f-exp> <f-exp>
```

where `<f-exp>` can be a number, a fluent or an expression of fluents. The case in which the the condition is simply a comparison between one numeric fluent and a number constant is similar to the pure propositional case, because this condition depends only on the value of the fluent. A more interesting case is when the condition contains an expression including a fluent changed by TIFs and some other fluents controlled by actions. In particular, if we have an invariant constraint of the form:

```
(> (x) (lb))
```

where `(lb)` is set by TIFs, and `(x)` can be modified by actions then, multiple TIFs might generate threats to the constraint throughout the plan. We call *Bounded Trajectory Management Problems* problems with this kind of constraint structure.

4.2 Formalisation of the Bounded Trajectory Management Problems

In this section we provide a formalisation of a particular class of problems that arise when introducing TIFs.

Firstly, we introduce a number of definitions which support the class of problems that we address.

Definition 9 (Uncontrollable Numeric Fluent) *A numeric fluent is called* uncontrollable *if it appears in Timed Initial Fluents, but is not modified by any effects of actions.*

In both the voltage control and UCP domains, the consumers' demand is an uncontrollable numeric fluent. The demand is fixed and it varies with time, and it is independent of any action taken to modify the voltage control problem.

Definition 10 (Controllable Numeric Fluent) *A numeric fluent is called* controllable *if it is modified by the effects of actions but not by any TIFs.*

If we consider the UCP problem, the generation output is a controllable numeric fluent, because it is completely determined by the actions that are chosen by the planner. Dividing the fluents into these two categories we do not consider fluents that are modified both by actions and by TIFs. However, we consider expressions with both controllable and uncontrollable numeric fluents, as defined in the following;

Definition 11 (Mixed Metric Expression) *A* mixed metric expression *is a function of terms in which both controllable and uncontrollable numeric fluents appear.*

Mixed metric expressions can also be used to indirectly model relative changes of numeric fluents with TIFs, for example of the form `(at 5 (increase (x) 5))`. Although we do not have the syntax to model this kind of event, it is always possible to decompose the fluent (x) into two fluents, respectively uncontrollable `(y)` and controllable fluents `(z)`, and replace the original fluents with the mixed metric expression (e.g `(+ (y) (z))`). Using the TIF `(at 5 (= (y) 5))` allow us to model the relative increase of `(x)`.

When a mixed metric expression is used in a comparison we have:

Definition 12 (Mixed Metric Constraint) *A* mixed metric constraint *is a comparison of numeric values in which both controllable and uncontrollable numeric fluents appear.*

Any mixed metric constraint can be written as a comparison between a mixed metric expression and a constant.

We now can define the Bounded Trajectory Management Problem as follows:

Definition 13 (Universal Bounded Trajectory Management Problem) *An* universal bounded trajectory management *problem (UBTMP) is a planning problem in which all metric fluents are controllable or uncontrollable and there is a single trajectory constraint of the form:*

$$(always\,(and\,P_1\,P_2 ... P_n))$$

where each P_i is a mixed metric constraint specifying an upper or lower bound on a mixed metric expression.

One way of specifying the goal of a UBTMP is to define a time horizon, and require the planner to construct a valid trajectory of states that will maintain the *always* condition (i.e. maintain the fluents within their bounds) up to that horizon.
We can generalise the UBTMP for cases with a trajectory that does not have to be true for the entire plan execution, but it must hold during an extended period of time. We call this class of problems *localised bounded trajectory management problems.*

Definition 14 (Localised Bounded Trajectory Management Problem) *A* localised bounded trajectory management problem *(LBTMP) is a planning problem in which all metric fluents are controllable or uncontrollable and there are invariant constraints (*flexible trajectory constraints*) c_i of the form:*

$$(hold\text{-}during\ t_{i,min}\ t_{i,max}(and\,P_1\,P_2 ... P_n))$$

$\forall t$ in $(t_{i,min}, t_{i,max})$, where P_i is a mixed metric constraint specifying an upper or lower bound on a mixed metric expression.

The constraints c_i must hold at some point during the plan, but their start and end points need not coincide with the start and the end of the plan. These problems model constraints of the type `(hold-after t P)` or `(hold-during t t' P)`, for some mixed metric constraint P and times t, t' that we choose. This is particularly useful when the initial state does not satisfy the constraints, but we give the planner time to react to the situation by finding a state that satisfies the constraints before the start of the trajectory. LBTMPs are a generalisation of UBTMPs. When referring to problems in either class we will call them BTMPs *(Bounded Trajectory Management Problems)*.

Many problems can be seen as BTMPs. For example, managing the speed of a vehicle so that it remains within safe speed limits, managing the temperature of a room or building so that it remains within a comfortable range and, as in our motivating problem, managing network voltage so that it stays within safe bounds. These problems can also be treated as control problems (Maciejowski, 2002), for example: thermostatically controlling heating and cooling systems to maintain room temperature. These problems are interesting for planning if the control actions take a significant amount of time relative to the variations generated by the background processes (or, equivalently, if the control system is slow to react to changes in the background processes), or if the BTMP appears in the context of additional goals that require discrete causal reasoning alongside the management of the trajectory constraints.

We can recognise two sub-classes of BTMPs: those in which no actions, or durative actions, have mixed metric constraints as conditions and those in which at least one action does have such a condition. In this thesis we focus our attention on the second sub-classe.

The definition of BTMPs allows for any combination of PDDL2.2 features, including TILs, TIFs, duration constraints and *over all* constraints. Any problem that contains background numeric curves expressed using TIFs, and trajectory constraints expressed as an *always*, *hold-after* or *hold-during* constraints is a BTMP.

Modelling Trajectory Constraints in PDDL2.2.

The PDDL3 language (Gerevini et al., 2009) supports the specification of trajectory constraints such as `(always P)`, `(sometimes P)`, `(hold-after t P)` etc. However, no other variant of PDDL supports this feature directly, and the planner POPF2 does not support PDDL3, so such constraints need to be compiled into other constructs.

The `(always P)` constraint can be modelled by constructing a durative action which enforces P as an *over all* constraint. Since P must be true over the whole plan trajectory (at least, the part of the trajectory where actions could affect the satisfaction of the constraint), it is essential that this action which, following Fox *et al.* (Fox et al., 2004) we call an *envelope* action, starts before any other action in the plan (including TIFs), and ends after every other action. To ensure that the envelope starts at the beginning of the plan it must have, as an *at start* precondition, a proposition that is provided in the initial state and removed by a TIL at time 2ϵ. This gives just enough time for the envelope to start before the condition is deleted. The envelope action must have a start effect that asserts a proposition that is specified as a precondition of every other action, to prevent other actions starting before the envelope opens. According to the PDDL2.1 semantics, the *over all* condition must hold within the open interval between the start and the end of the action. However, the constraint is guaranteed to be satisfied for the entire duration of the plan because of the *no moving*

Initial state:

```
(dummy)
(at 0.002 (not (dummy)))

(= (x) 5)
(= (y) 5)
(= (lb) 0)
(= (up) 10)
(at 10 (= (y)  10))
(at 20 (= (y) 15))
(at 10 (r))
```

Goal

```
(done)
```

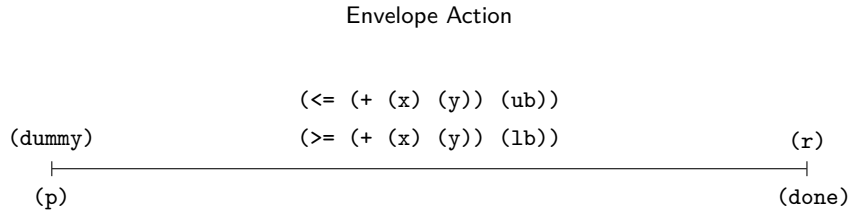


Figure 4.4: The result of translating `(always (and (>= (+ (x) (y)) (lb)) (<= (+ (x) (y)) (ub))))` into an envelope action.

target rule and the fact that the proposition is deleted at 2ϵ. These means that no action can change the fluents in the constraints before the open interval (Fox et al., 2004). Finally, the end of the envelope must not occur until the goal of the problem is achieved. This might be at the time horizon, if specified, or when some condition becomes available. This condition must correspond to the fact that the goal is achieved. Figure 4.4 shows how this translation is done.

It might seem sufficient to require that the envelope contains all actions of the plan, without forcing it to start at the beginning of the plan timeline. However, if the planner can delay the start of the envelope until after the TIFs, it will be able to avoid the *always* constraint interacting with all but the last TIF.

Similarly, for the *hold-after* and *hold-during* constraints an *envelope action* can be similarly modelled. The only difference with respect to the *always* constraint is when the *at start* or *at end* preconditions are deleted or added by TILs.

Simple Example of BTMP

A simple example of a BTMP is the *Simple Unit Commitment Problem*. In this domain, an electrical demand must be satisfied over a time period. Since we do not want to waste too much energy, the supplied power generated cannot exceed an upper bound that depends on the demand. During that period of time there are a number of TIFs representing the uncontrollable discretised demand curve. The demand can be satisfied by supplying electrical energy generated by some generation units. When a generation unit is switched on, it is possible to ramp-up the unit to produce more or less energy. These actions, reported in Figure 4.5, have a fixed duration to model the fact that the ramping has a maximum rate that cannot be exceeded. These actions are also identified as compression safe actions, by POPF-TIF. In this simple domain we assume that there is only one generation unit and this is always on. The goal is to keep the supply inside the envelope defined by the bounds. To

```
(define (domain simpleUnitCommitment)
    (:requirements :typing :fluents
         :durative-actions
         :timed-initial-literals
         :negative-preconditions)
    (:predicates
         (r) (done) (q)
         (can-ramp))
     (:functions
          (demand) (supply))

    (:durative-action ramp-up
         :parameters ()
         :duration (= ?duration 1)
         :condition (and
             (at start (can-ramp)))
         :effect (and (at end (can-ramp))
             (at start (not (can-ramp)))
    (at start (increase (supply) 10))))

    (:durative-action ramp-down
         :parameters ()
         :duration (= ?duration 1)
         :condition (and
             (at start (can-ramp)))
         :effect (and
             (at start (not (can-ramp)))
    (at end (can-ramp))
    (at start (decrease (supply) 10))))

    (:durative-action envelope
         :parameters ()
         :duration (<= ?duration 100000)
:condition (and (at end (r))
    (over all (>= (supply) (demand)))
    (over all (<= (supply) (+ (demand) 20)))
    (at start (q))
    )
:effect (and
    (at end (done)))))
```

```
(define (problem simpleProblem)
    (:domain simpleUnitCommitment)
    (:init
         (can-ramp)
         (= (supply) 50)
         (q)
         (at 0.005 (not (q)))
         (at 50 (r))
         (= (demand) 50)
         (at 20 (= (demand) 60))
         (at 40 (= (demand) 75))
         )
     (:goal (and (done)))
)
```

Figure 4.5: The Simple UCP domain and an example problem instance.

guarantee the satisfaction of the constraints, we use the translation defined in the previous section. Thus, we add an envelope action, with the invariant constraint, to the domain. Figure 4.6 shows the kind of plan that we hope to build automatically using a planner.

It is easy to confirm that this problem is a BTMP: the `demand` variable is an uncontrollable variable, while `supply` is a controllable variable. The mixed metric constraints in the problem are:

```
(>= (supply) (demand))
```

and

```
(<= (supply) (+ (demand) 20)))
```

Figure 4.7 shows the profile of the background demand, which increases above the

```
0: (envelope) [50]
0: (ramp-up) [1]
20: (ramp-up) [1]
21: (ramp-up) [1]
```

Figure 4.6: A valid plan for the problem shown in Figure 4.5.

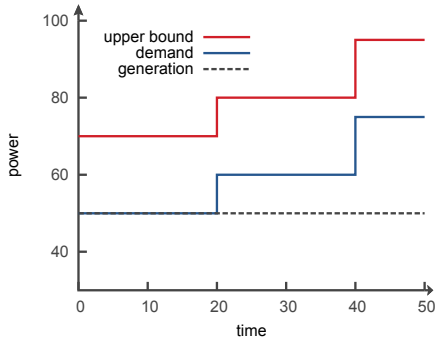


Figure 4.7: Initial state for the problem shown in Figure 4.5.

supplied energy, requiring the plan to switch on the generator and ramp-up at least three times to meet the demand.

4.3 Challenges of BTMPs

In this section we consider the challenges raised by BTMPs in a forward heuristic search framework and in particular in POPF-TIF. The problem is that each TIF can be a threat to the trajectory constraints and this is not seen by the planner until it applies the TIF. We explain this concept using our running example.

In the instance of the Simple UCP domain presented in Figure 4.5, Section 4.2, the demand is initialised to a value that lies within the specified bounds, but it is then changed by TIFs. Once the constraint envelope action is started, the mixed metric constraint becomes active. If POPF2 evaluates a state in which the constraint is not satisfied then the state must be considered a dead-end (by definition, a state must satisfy the condition to be valid). The monotonicity of the relaxed reachability construction will ensure that the condition remains satisfied throughout the TRPG constructed from any state in which the condition is already satisfied. Thus, the relaxed plan will include no actions to react to TIFs and the heuristic will not see the cost of responding to future TIFs.

In the Simple UCP problem, the heuristic distance from any state to the goal is therefore given by the number of TIFs and TILs that remain to be executed, as shown in Figure 4.8. No actions are needed and the only helpful actions proposed are the application of the next TIL or TIF. The consequence of this is that the planner steps forward past the next TIL or TIF and only discovers the violation of the *always* constraint as a result of progressing the state into one that fails to satisfy the condition. EHC fails immediately and the planner performs a best-first search of the search space. The Simple UCP problem is very small, so this behaviour still allows a plan to be found. As is seen in Figure 4.9, this leads to 21 states being explored. Examination of the search tree shows that the planner never sees the next TIF until it is reached, so it cannot distinguish between the relative values of ramping up and ramping down the units. Indeed, the heuristic values of states generated using increase actions are the same as those generated using decrease actions, because the effect on the state in either case is equivalent under the relaxation. To solve this particular Simple UCP problem, the planner has to increase the supply quite aggressively (three successive increases

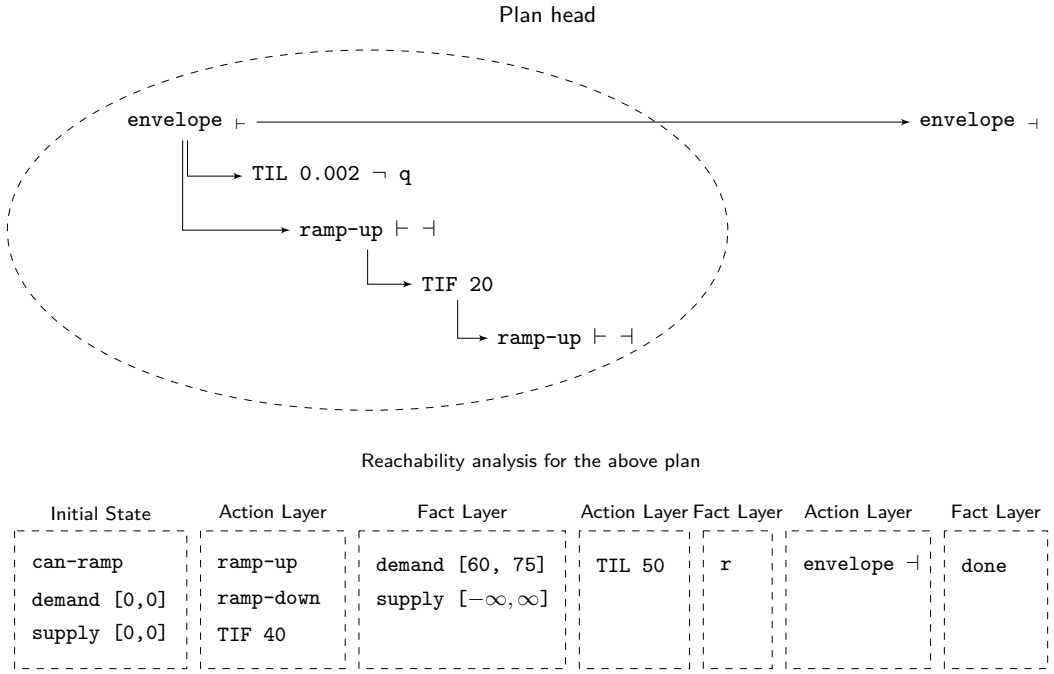


Figure 4.8: The plan head is used to construct the initial state and the first layer of applicable actions includes all actions that could be applied from the initial state, allowing time to pass if necessary

are required), but it has to search its way to this solution by backtracking out of bounds violations caused by increasing voltage instead. The key issue is that the heuristic provides no guidance and it does not see starting a ramp-up action as helpful.

4.4 Lookahead Heuristic

In pursuing a more informative heuristic, we need to take into account the effects of TIFs during the heuristic evaluation. Keyder et al. (2012) observed that the FF relaxed plan heuristic sometimes oversimplifies the structure of a problem, resulting in an inaccuracy of the heuristic value. By *de-relaxing* the problem we do not ignore all the constraints as in the standard relaxation, but we retain some of the original constraints to gain more information.

It might be thought that treating TIFs in the same way as TILs, during the construction of the TRPG, would increase the heuristic information available. Using this approach, instead of widening the interval of accessible system voltages, the TIF effect would *reduce* the interval to the range of numeric values actually achievable at each time-point. No actions depending on values outside this range would be applicable after that layer, although they would continue to be applicable if they had been applicable earlier, in order to retain monotonicity. While this narrowing of the TRPG at each TIF layer might provide additional heuristic information for some problems, by leading to longer relaxed plans, it does not help for BTMPs in which there are no mixed metric constraints in the preconditions of actions, since in this case the reachable actions are not influenced by the uncontrollable variables and the relaxed plan therefore cannot be affected either.

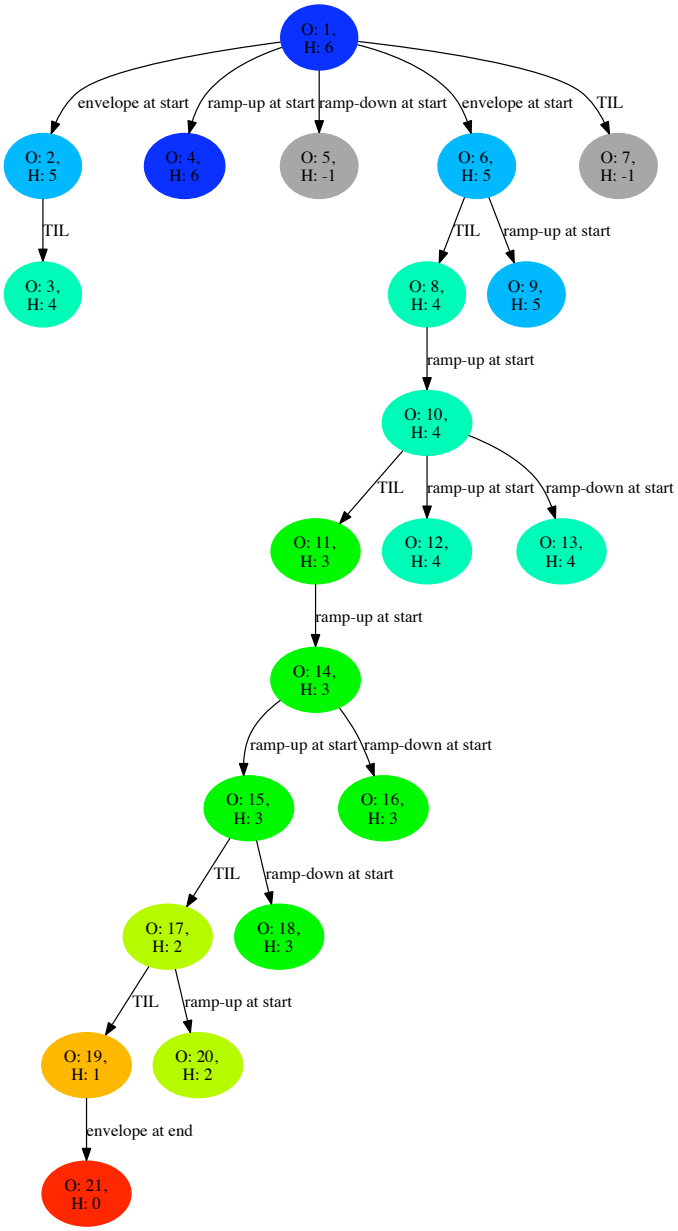


Figure 4.9: The search space explored in solving the Simple UCP problem using standard POPF2 search. 21 states are explored. The states are labelled with the order number in which they are visited (O) and their heuristic values (H).

What we do, instead, is to anticipate the effects of TIFs in the heuristic evaluation, so that the threads to the *over all* constraints can be repaired in the RPG. We call this method *lookahead heuristic*. Before providing details of the *single lookahead heuristic* and its extension *multiple lookahead heuristic*, we provide some definitions.

Definition 15 *(*Active over all constraint*) An over all constraint* c=`(over all` P`)`*, in action* a*, is* active *in a plan state* s*, with respect to a TIF* `(at` t `(= (v)` α`))` *if* P *depends on* v*,* $a_{\vdash}$ *is in the plan head in* s*,* $a_{\dashv}$ *is not in the plan head and does not appear in the TRPG before layer* t*.*

Definition 16 *(*TIF-grounded goal*) If* c=`(over all` P`)` *is an active constraint with respect to a TIF* T=`(at` t `(= (v)` α`))`*, a TIF-grounded goal for* T *is* $P[v/\alpha]$ *at* t*.*

4.4.1 Single Lookahead Heuristic

To improve the heuristic for search guidance in BTMPs, the TRPG is constructed in exactly the same way as in POPF2, but we modify the relaxed plan extraction phase. In POPF-TIF, in addition to the goals in the final layer of the TRPG (representing the final goals of the planning problem), goals are set at the earliest layer in which the first TIF effects are added: for each active *over all* condition that includes a mixed metric constraint referring to one of the fluents modified by a TIF in this layer, the constraint is added as a goal, substituting the newly assigned TIF value into the constraint. For example, in the Simple UCP problem, when evaluating the state following the application of the start of the envelope action, the first TIF is `(at 20 (= (demand) 60))`. Two active constraints are present. Therefore, two TIF-grounded goals are added at time 20, formed by evaluating these active constraints with `demand` set to 60. These are:

`(>= (supply) (60))` and

`(<= (supply) (80))`

More generally, when evaluating a state, TIF-grounded goals are added at the first time-point at which TIFs appear in the TRPG, corresponding to all active constraints in the state being evaluated with respect to those TIFs.

These goals are managed in the same way as any other preconditions of actions added during the relaxed plan extraction phase: new actions are added to achieve them and counted in the heuristic. In this example, these goals will lead to the addition of one `ramp-up` start snap-action in the initial layer of the plan (followed by its end snap-action at time 1), to ensure that the supply is increased above 60 (the lower-bound). The heuristic value is then the sum of the number of remaining TIFs and TILs together with the start of the `ramp-up` action, which is also a helpful action.

An effect of this mechanism in general is to both enhance the heuristic value by accounting for actions required to meet the *over all* conditions and also to ensure that *helpful actions* are added to achieve these goals which are essential for the planner to avoid the violations caused by the TIFs. Without this modification, the relaxed plan would not contain these helpful actions and the planner would often choose the wrong action to apply. This mechanism can be seen as a *lookahead* over the upcoming TIFs, enabling the heuristic to identify the actions that will be helpful in avoiding the violations they might cause.

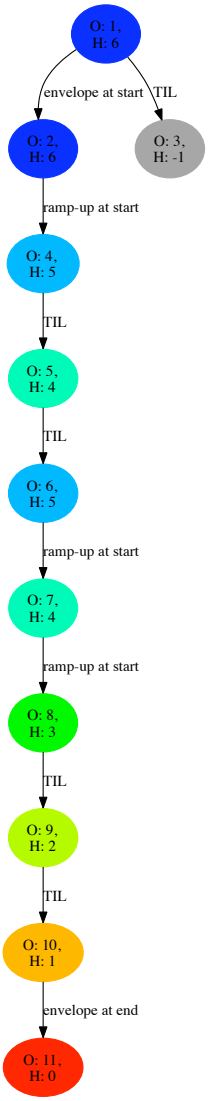


Figure 4.10: The search space explored in solving the Simple UCP problem with a lookahead of 1. Only 11 states are explored.

As can be seen in Figure 4.10, this mechanism has a significant effect on the search behaviour of the planner. Our Simple UCP problem instance is solved visiting just 11 states. Looking in detail at the search tree, it can be seen that the EHC encounters a plateau at the outset (where the heuristic value of the initial state is 6, but its successors are a state of value 6 and a dead-end state). The search crosses this successfully when it encounters state 4, with value 5. Thereafter, there is no further search (although the heuristic does increase as the next TIF is encountered). We can see that the heuristic has identified the `ramp-up` actions as being the helpful ones. The plan first starts the check action, which is our *always*-enforcing envelope action, then a TIL closes the opportunity for this by deleting the envelope precondition. Next, the plan starts, and then ends, a `ramp-up` action, in order to increase the supply to avoid a violation when the first TIF occurs. Two further `ramp-up` actions are started before the next TIF, to avoid the anticipated violation. Finally, a TIL asserts the end condition of the check action, so the check-end is applied and the goal state is reached.

4.4.2 Multiple Lookahead Heuristic

The extension of the lookahead mechanism to n-lookahead (*multiple lookahead*) is straightforward: the first n TIFs added to the reachability analysis are compared with active *over all* constraints and, where they would cause a violation of a constraint, the corresponding TIF-grounded goal is added to the layer in which the TIF is applied.

This extension would benefit in situations in which the actions that repair the active trajectory constraints are not applicable before the first TIF occurs, but they need more time to apply the effects. In this case the single lookahead only detects the dead-end caused by the next TIF, but it fails in identifying the helpful actions, because there are no more that are applicable at this point in the plan. In contrast, looking ahead over further TIFs can detect early the dead-end state caused by a TIF and therefore identify additional helpful actions and contribute to a more informed heuristic value.

Although this lookahead mechanism adds to the computational cost of the heuristic evaluation, computation of the heuristic remains polynomial, since it adds at most m goals to the relaxed plan graph, where m is the number of constraints of the open actions that are violated.

4.5 Violation of Trajectory Constraints

For completeness we consider an alternative way to deal with violations of trajectory constraints by allowing them if they happen at a granularity lower than the responsive time of the system.

TIFs can be used to describe the predictable behaviour of continuous functions, although a discretisation must be provided by the model designer, since our syntax for TIFs allows only to specify the value of functions at specific time-points. The granularity of the discretisation of the function depends only on the specific problem and it is set in the initial state. An example of discretisation is shown in Figure 4.11.

The presence of the discretisation to approximate a continuous function in a BTMP can

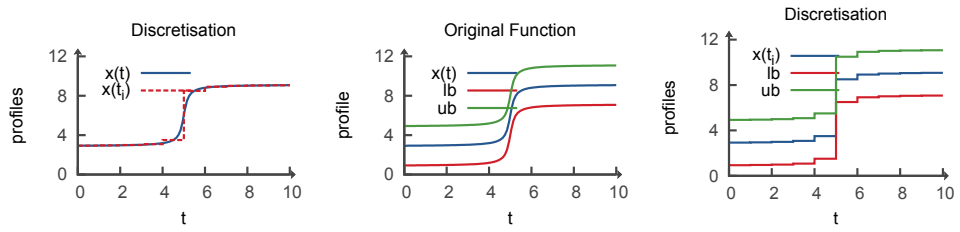


Figure 4.11: Example of a function and its discretisation with a granularity of 1 unit of time.

create some conflicts with the *no moving targets rule*. Let us consider the following example: we have an uncontrollable fluent `(x)` and a controllable fluent `(y)`, which take part in the mixed metric constraints:

$$\begin{aligned} \mathtt{y}(t) &< \mathtt{x}(t) + a, \\ \mathtt{y}(t) &> \mathtt{x}(t) - b. \end{aligned} \tag{4.1}$$

for some constant parameters $a,\ b > 0$ and for each $t \in (t_0, t_1)$. Figure 4.11 shows this example. These constraints are *always* constraints of a UBTMP if $t_0 = 0$ and $t_1 = \infty$. If the uncontrollable variable `(x)` is an approximation of a continuous function, it might happen that the discretisation is not fine enough and we end up having

$$\mathtt{x}(t_{i-1}) + a < \mathtt{x}(t_i) - b, \tag{4.2}$$

for some point t_i. This is the case shown in Figure 4.11, where at time 5, the lower bound becomes higher than the upper bound at time 4. In this case it is evident that no actions that increase the value of `(x)` can be applied before or after $t = 5$ without violating the constraints. An instantaneous action could be applied exactly at time $t = 5$, but this is not possible without violating the *no moving target* rule, so the planner rejects it. However, in practical the situation, the action might be actually applicable, but we cannot see it because of the approximation that we have used to model the original function.

To overcome this problem there are two possible solutions. A first method is based on re-writing the function $\mathtt{x}(t)$ using a time-step that takes into account the variation of the curve and the a and b parameters of Equation 4.1. Another approach is to recognise that a violation of the constraint is acceptable, provided that it lasts for a short enough time with respect to the discretisation of the initial function.

4.5.1 Rewrite the Discretisation

The first approach that we consider is to write a different approximation of the initial function. One can consider a finer discretisation, or alternatively a discretisation with non constant time-step, for example choosing the width of the step ($p = t_i - t_{i-1}$) such that:

$$\mathtt{x}(t_{i-1}) + a = \mathtt{x}(t_i) \ \vee \ \mathtt{x}(t_{i-1}) - b = \mathtt{x}(t_i) \tag{4.3}$$

In this way there is an interval of time in which an action can be applied without violating

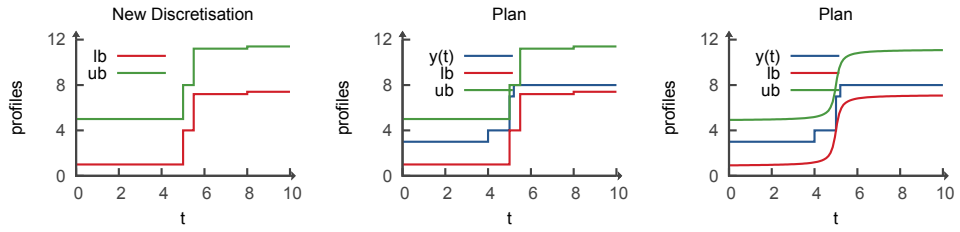


Figure 4.12: Example of the new discretisation and the resulting plan.

the constraint in Equation 4.1. An example of the solution of this domain is shown in Figure 4.12.

4.5.2 Allow a Small Violation of Condition

A second approach to the problem of the discretisation of the background function is based on the assumption that the granularity of discretisation of the function $\mathtt{x}(t)$ is the minimum time in which effects on the system can be observed. This means than events that happen within the time step can be considered "almost instantaneous" and a violation of the trajectory constraints is not seen by the system in practice. For this reason the planner might accept actions that violate the trajectory condition, provided that the violation lasts for a small enough period of time.

There are two methods to allow the violation of the condition. The first one is to "manually" increment the parameters a and b of Equation 4.1 to arbitrary big values, immediately after the discretization of the function $x(t)$ and restore the original values before the next TIF is applied.

Alternatively, instead of modifying the problem file in PDDL, it is also possible to obtain the same results modifying the planner in order to accept actions that violate preconditions and imposing that these will be restored $c \cdot \epsilon$ times later. Requiring the restoration to happen only an ϵ time later might be too restrictive in cases where multiple actions are needed. In these cases, we need to define a constant c that determines the tolerance of the violation.

This new functionality can be implemented in the planner POPF-TIF. Instead of rejecting a happening a_0 (an actual action or a TIF) that violates a trajectory constraint, POPF-TIF can mark the violating happening. In addition it adds as a goal the precondition violated. The sequence of actions $a_1, ..., a_n$ that achieve the new goal are than added to the plan with the additional ordering constraints:

$$t_n - t_1 < c \cdot \epsilon \tag{4.4}$$

where t_1 and t_n are the minimum time-stamps of the happenings a_1 and a_n respectively, ϵ is the minimum separation time of two actions and c is a constant number such that $c \cdot \epsilon$ is less than the discretisation time of the function $\mathtt{x}(t)$.

An example of allowed violation is shown in Figure 4.13.

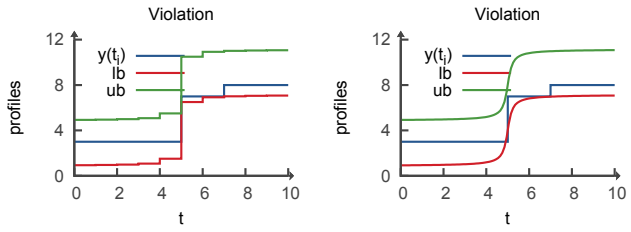


Figure 4.13: Example of violation of *over all* condition and the resulting plan. The violation occurs at time 5, but it is resolved at time 5.001.

4.5.3 Comparison of the Two Approaches

As shown in Figure 4.12, the first approach produces solutions closer to the actual function boundaries. However, this method requires work in the pre-processing stage and the *a priori* knowledge of $\mathtt{x}(t)$. This function cannot always be represented with an analytic form and can also be subject to changes by other actions present in the domain, making this method not always applicable.

On the contrary, the second method can always be applied, but there must be a solid motivation to allow violations of constraints. This can be the case when the period of violation $c \cdot \epsilon$ is not observable by the physical system considered.

The first method is preferable when there is a complete knowledge of the functions $x_i(t)$, whilst the second is more appropriate in situations in which the granularity of the functions is determined by physical motivations.

Although we experimented with these ideas, we decided not to explore them further to avoid having two distinct time granularities, once for the planner and one for the physical system that we are considering.

4.6 Alternative Search Algorithm

In POPF-TIF, TIFs and TILs are considered as dummy actions that must be applied at their specified time points.

We have the following definition:

Definition 17 (TIF state) *A* TIF state *is a state where the last action inserted in the plan is a TIF.*

In BTMPs, TIF states are potentially temporal dead-ends. In POPF2, the standard search strategy is to apply EHC until it fails, and then to perform a best first search from the root of the search space. This means that the plan head generated up to the TIF causing the dead-end is lost and it needs to be reconstructed at an high computational cost. Moreover, once the TIF that caused the failure of EHC is inserted into the plan, the search continues to exhaustively explore the search space, until a goal state is found.

The advantage of using EHC is that we can find a solution more quickly than best first search, exploring a relatively small number of states. In addition, EHC exploits helpful action pruning to further reduce the number of states evaluated. The drawback is that EHC might fail to find a solution even if one exists. In contrast, BFS is a complete algorithm.

Algorithm 1: Mixed Search Algorithms

```
Data: Planning problem
Result: Plan
S_0 ← initial state;
backupQueue = {};
searchQueue = {initial state};
t_max = 0;
while run EHC from searchQueue do
    S ← state selected from searchQueue;
    updated t_max;
    if S is a TIF state then
        insert S in backupQueue;
    if EHC fails then
        while backupQueue is empty do
            S_b ← selectNodes ;
            while run BFS from S_b do
                if S_b is a TIF state and t_{S_b} is t_max then
                    update Plan;
                    update searchQueue;
                    restore EHC;
```

Algorithm 2: Select Node for Mixed Search Strategy 1 (MS1)

```
Data: Planning problem
Result: queue
S_b ← backupQueue.front() ;
```

Thus, if we give it enough time and memory, it will find a solution, but exploring many more states. In order to speed up the search, exploiting EHC as much as we can, while not loosing the backtrack capability of the BFS, we design two mixed search algorithms that combine these two searches strategies, decomposing the search around the problematic TIFs.

The two mixed search strategies (MS) (Algorithm 1) make use of two parallel search queues. The first one (*searchQueue*) is initialised to the initial state, while the second queue *backupQueue* is initially empty. In addition, the real variable t_{max} is set to the value 0. The *searchQueue* is used to perform an EHC search. During the search, TIF states are inserted in the *backupQueue*. The variable t_{max} is updated to the value of time at which the next TIF will be inserted. If EHC fails to find a solution, instead of resorting to BFS from the initial state, the MS strategies resort to BFS from states selected from the *backupQueue*. The BFS terminates if it finds a solution or if adds a TIF state where the time of the applied TIF is equal to t_{max}. In this second case, EHC is restarted starting from a new *searchQueue* initialised to the last state found by the BFS. If the BFS fails, then new nodes from the *backupQueue* are selected to be inserted in the *searchQueue* of a new BFS.

The two algorithms differ from each other by how the states are selected from the *backupQueue* to be inserted in the starting *searchQueue* of the BFS. The first Mixed Search algorithm (MS1) selects the node that is most recently added in the *backupQueue*, as shown in Algorithm 2. This search strategy rests on a strong assumption that the resolution of

Algorithm 3: Select Node for Mixed Search Strategy 2 (MS2)

Data: Planning problem
Result: queue
1 $S_b \leftarrow backupQueue$;

each TIF is independent of the plan preceding the last TIF. The second strategy is to resort, on failure of EHC, to a TIF state earlier than the failure point, but not necessarily the most recent. This is achieved by starting the BFS with the entire *backupQueue*, so that the BFS can backtrack to whichever TIF state appears most promising, as shown in Algorithm 3. This method is referred to as Mixed Search Algorithm 2 (MS2). In Figure 4.14 the search space explored by these two search algorithms is represented for the simple illustrative example. In this case, the BFS is called in two points to resolve the problem created by the first and the last TIFs. The nodes in the box are the ones explored by BFS. In this example the two search algorithms explore the same states.

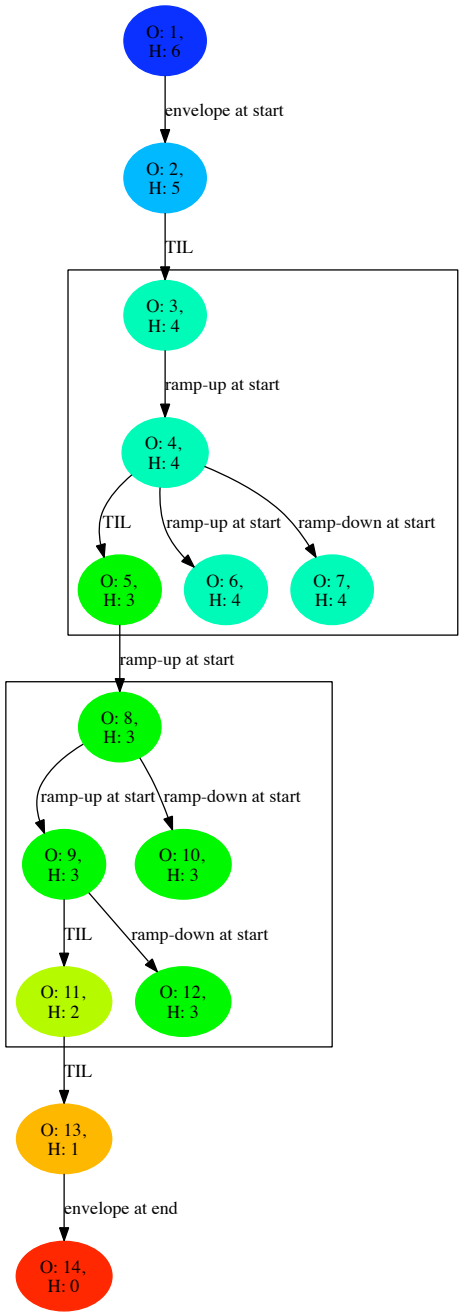


Figure 4.14: The search space explored in solving the Simple UCP problem with the mixed search algorithms. Only 14 states are explored.

Chapter 5

Semantic Attachment

In this chapter we present our approach to the integration of an external solver with a planner capable of solving BTMPs. Others have attempted this sort of integration, calling it semantic attachments. We start by giving the motivation for the need for a different approach and by presenting the formalism to understand our approach. We compare our method with previous approaches in the literature and we describe our way to achieve integration by means of *encapsulated types*. We then provide the details of the implementation of the coupling between the planner POPF-TIF and an external solver that manages the external evaluated functions of *encapsulated types*.

5.1 Introduction and Motivations

When considering the AC voltage control problem, an issue introduced by the presence of the power flow equations is that the effects of actions propagate throughout the network, changing the voltages at every busbar. Typically, to model an action, every possible effect that the action will bring about has to be specified. Instead, having to deal with power flow equations makes it difficult to express all the effects on every element of the network. However, it is possible to recognise two types of effects: *direct*, and *indirect* effects. The direct effects are the ones that the planner can bring about by the application of actions, while the indirect effects are those brought about by the environment in which the action is applied, as a *consequence* of the application of the action. This problem is known in the AI community as the *ramification problem* (Sandewall, 1994).

We clarify the role of the *direct* and *indirect* effects with an example. In the paper (Löhr et al., 2012) on hybrid planning, an example with a *maze ball* was presented. In this domain there is a platform that can be inclined through fixed angles. A ball rolls around on it in a way that is determined by the inclinations of the platform. The planning task is to move the ball to a destination position by performing multiple inclinations of the table. The movements of the table are the direct effects of the inclinations chosen by the planner, while the movements of the ball are indirect effects. In the method described in the paper, the position of the ball is calculated after every inclination action and is treated as part of the direct effect of the action. Instead, with our approach, we distinguish the *direct* effect of inclining the table and the consequence that it causes on the movement of the ball. In the

same way, in the AC voltage domain we can separate the direct effects, such as changing the tap-settings of transformers, from the indirect ones, such as the resulting change of voltage. A crucial aspect of the *indirect* effects is that they often cannot be expressed with simple arithmetic formulas. Indeed, they might be complex to compute and it might not be possible to express the necessary functions in PDDL. In the AC voltage domain the relationship between direct and indirect effects is not trivial and the indirect effects cannot be expressed in standard PDDL because they require the use of numeric axioms and these are not part of the language.

As mentioned before, the *maze ball* domain can be seen as an example of one with *indirect effects*. In this case, the only indirect effect caused by the inclination of the platform is the variation of the position, velocity and acceleration of the ball. Since the number of *indirect effects* is limited, they can be merged with the *direct* ones. The link between them is solved in a pre-processing stage, so that they can be written as a look-up table in the domain. In contrast, our domain has a large number of objects which interact with each others and a similar approach is not suitable because the dimension of the look-up table blows-up exponentially with the size of the network.

In the PSR domain (Thiébaux and Cordier, 2001), the link between the *direct* and *indirect* effects, which are purely propositional, is made by the use of derived predicates (Thiébaux et al., 2005) in PDDL2.2 (Hoffmann and Edelkamp, 2005), while no numerical equivalent to derived predicates is formalised in the language or considered in the implementation of any planner.

There are other works in AI planning focused on extending planning to handle more complex problems. Dornhege *et al.* (2009) extended PDDL to include grounded predicates or changes of fluents evaluated by externally specified functions, as it will be explained in details in Section 5.3.3. A more general approach is Planning Modulo Theories (PMT) (Gregory et al., 2012), which is an extension of planning to support first order theories as parameters. Although in our approach we do not offer a general interface to external solvers or provide general methods for supporting theories and their corresponding types, our external module is not considered as a complete "black box" called during the evalaution of a state, but the planner communicates with it also during the heuristic evaluation, guiding the search in a more informative way.

5.2 Formalism

In this section we present a formalism for handling direct and indirect effects in the context of temporal planning.

To distinguish the direct and the indirect effects of the action, we divide the variables of a state in three categories:

- $V^{special}$: variables that record the indirect effects;
- V^{dep}: variables that are affected by actions, resulting in the external update of $V^{special}$ variables;
- V^{indep}: variables that are affected by actions without resulting in any external computation.

Definition 18 *Given the three categories of variables defined above, the state can be written as:*

$$s = \langle v_1^{dep}, ..., v_{n_1}^{dep}, v_1^{indep}, ..., v_{n_2}^{indep}, v_1^{special}, ..., v_{n_3}^{special} \rangle \in \vec{V}$$

The PDDL actions can affect only the variables in $V^{dep} \cup V^{indep}$, while goals and conditions can be expressed in terms of all of the variables in $V^{dep} \cup V^{indep} \cup V^{special}$. Whenever a change in at least one of the V^{dep} variables occurs, the effects are propagated to the $V^{special}$ variables. For example: in the AC voltage control domain, the V^{dep} variables include the tap settings which are changed when the planner applies an action to step a transformer up (or down). The $V^{special}$ variables include the busbar voltages.

We identify two types of transition functions, one describing the effects of actions and the other describing the resulting indirect effects.

Definition 19 *The effects of an action, a, are represented by a function:*

$$\mathit{effect}(a) : \vec{V} \rightarrow \vec{V}$$

such that

$$\begin{aligned} \mathit{effect}(a)(s) &= \mathit{effect}(a)\left(\langle v_1^{dep}, ..., v_{n_1}^{dep}, v_1^{indep}, ..., v_{n_2}^{indep}, v_1^{special}, ..., v_{n_3}^{special} \rangle\right) \\ &= \langle v_1^{'dep}, ..., v_{n_1}^{'dep}, v_1^{'indep}, ..., v_{n_2}^{'indep}, v_1^{special}, ..., v_{n_3}^{special} \rangle \end{aligned}$$

so that the effects of the action are restricted to the variables $V^{dep} \cup V^{indep}$.

Definition 20 *An indirect effect ϕ is a function:*

$$\phi : \vec{V}^{dep} \rightarrow \vec{V}^{special}$$

such that

$$\phi\left(\langle v_1^{'dep}, ..., v_{n_1}^{'dep} \rangle\right) = \langle v_1^{'special}, ..., v_{n_3}^{'special} \rangle$$

Note that indirect effects are a special case of events (Fox and Long, 2006). In general, events are effects that contribute to change a state into another, but in contrast to the actions, the planner cannot control them. In the case of indirect effects, these events are triggered by a change of the V^{dep} variables.

Definition 21 *An instantaneous state transition triggered by action a, $\mathit{effect}(a)_\phi$, is defined as a function:*

$$\mathit{effect}(a)_\phi : \vec{V} \rightarrow \vec{V}$$

such that

$$\begin{aligned} \mathit{effect}(a)_\phi(s) &= \langle v_1^{'dep}, ..., v_{n_1}^{'dep}, v_1^{'indep}, ..., v_{n_2}^{'indep}, \phi\left(\langle v_1^{'dep}, ..., v_{n_1}^{'dep} \rangle\right) \rangle \\ &\quad \textit{where} \\ &\quad \langle v_1^{'dep}, ..., v_{n_1}^{'dep}, v_1^{'indep}, ..., v_{n_2}^{'indep}, v_1^{special}, ..., v_{n_3}^{special} \rangle = \mathit{effect}(a)(s) \end{aligned}$$

Whenever an action, a, is applied to a state, s, in a domain that includes an indirect effect function ϕ, s is updated by the corresponding instantaneous state transition effect$(a)_\phi(s)$.

Here, ϕ is the function that is implemented by the external solver and is therefore an external function, which needs a way to be modelled in the domain description.

5.3 Related Works

In the PDDL family of languages only a sub classes of indirect effects can be modelled. We give an overview in the following.

5.3.1 Derived Predicates in PDDL2.2

If we restrict to the case of having only propositional variables, then indirect effects can be modelled as derived predicates in PDDL2.2 (Hoffmann and Edelkamp, 2005).In this case the predicate symbols are divided into two subsets $\mathcal{B}$ and $\mathcal{D}$ (with $\mathcal{B} \cap \mathcal{D} = \emptyset$), the sets of basic and derived predicates respectively. The predicate symbols in $\mathcal{B}$ are part of the $V^{dep} \cup V^{indep}$ variables in Definition 18, while the subset $\mathcal{D}$ is equivalent to $V^{special}$. Symbols in $\mathcal{D}$ are not allowed to appear in the initial state description or in the effects of actions, but they only may appear in preconditions and goals. The truth values of the derived predicates are determined by a set of rules over the basic and the other derived predicates. In this sense derived predicates differ from Definition 20 because they map $V^{dep} \cup V^{special} \rightarrow V^{special}$. This allows cascade events that can occur sequentially. In PDDL2.2, however, a derived predicate cannot appear in negated form in any rules, avoiding negative interactions between applications of the rules.

As argued by Thiébaux *et al.* (2005), adding derived predicates to the PDDL language implies adding expressive power to the language. Although derived predicates can be compiled away, it is not always possible to construct compiled domain descriptions for which plan length increases only polynomially.

5.3.2 Events and Processes in PDDL+

The PDDL+ language (Fox and Long, 2006) was introduced to allow the planner to reason with effects that do not depend on the planner itself, but are triggered by the external environment. In this language events and processes are introduced, where the difference between the two lies in the type of effects. While events model instantaneous discrete effects, processes describe continuous changes of numeric variables. Events are defined as a generalisation of simple actions, where at least one numeric precondition must be present, while processes are similar to actions, but their effects are given only by a conjunction of process effects.

As in the case of derived predicates, and in contrast to Definition 20, processes and events can have preconditions given by any kinds of numeric variables, allowing cascade effects. In PDDL+ there is no distinction between V^{dep}, V^{indep} and $V^{special}$ variables, but numeric variables can be modified only by algebraic expressions of other quantities. This limits the kind of effects that can be modelled: simple manipulations of numeric quantities

such as the square root cannot be exploited in this language, as noted in (Parkinson et al., 2014).

5.3.3 Semantic Attachments in PDDL/M

Semantic Attachments where introduced in (Weyhrauch, 1980) as a way to interpret a predicate symbol using an external procedure. They were incorporated in PDDL with the name *module* by Dornhege *et al.* (2009), extending the PDDL language into PDDL/M. In this language, there are two type of modules, *condition-checkers* and *effect-applicators*. The former are defined by the predicate symbol and the external procedure that is required to calculate their values, while the latter are defined by the name and a list of numeric variables that need to be calculated by the external procedure. The actions can have in their preconditions condition-checker modules and a effect-applicator module as effects.

The advantage of defining modules is that the value of variables can be determined by not simple algebraic relations, but at the price that information on these variables cannot be easily embedded in every heuristic guidance. The work in (Dornhege et al., 2009) shows that only condition-checker modules can be integrated in the h^{FF} heuristic by calling the external procedures in a *faster* mode after every actions until the condition becomes true.

The modules in PDDL/M were not introduced in the context of direct and indirect effect, but external evaluated functions can be interpreted as indirect effects. Condition-checker modules are predicate symbols that must be checked after the application of actions, and they are similar to the derived predicates, with the difference that their value is determined externally. Effect-applicator modules, instead, modifies variables only after the application of determined actions. In this sense the condition-checker modules and the list of variables in the effect-applicator modules are the $V^{special}$ variables in Definition 18.

This approach is used to plan the operations of satellite observations (Aldinger and Löhr, 2013). The task is to determine the sequence of observation patches of a telescope, taking into account the slew manoeuvre of the satellite. In this problem the selection and the scheduling of the patches are performed by the planner TFD, while the computation of the slew manoeuvres are outsourced into the external modules.

In our domain this approach is not sufficient, because we need to be able to call the external module whenever a change happens, whether caused by actions or exogenous events. In addition, in their approach, during the heuristic evaluation, the information about the numeric fluents affected by an effect applicator is not exploited, as it is simply added in the relaxed reachability analysis, whenever the corresponding action is detected as reachable, regardless of the effects of the action. This leads to an over-relaxed approach in our domain.

5.3.4 Other Approaches in AI

The problem of combining planning with more complex effects of actions is also studied in other fields in AI, particularly in robotics. A particular class of problem that received much attention is the combination of task and motion planning. When performing different tasks, a robot has different level of reasoning. High-level reasoning is needed to find the sequence of actions to achieve a goal (*task planning*), while a lower-level reasoning is needed to plan for the physical execution (*motion planning*). This is a consequence of all the indirect effects

that the actions bring about and that are not modelled, but that can constrain the execution of actions.

In (Lagriffoul et al., 2013) the authors propose a way to combine task and motion planning building a constraint base approach that prunes the sequences of actions that are logically consistent but geometrically impossible to execute. Another work, (De Silva et al., 2013), links a hierarchical task planner to a geometric solver that computes geometric predicates. Similarly, the work (Srivastava et al., 2013) integrates a task planner with a spatial solver which rejects plans if some geometric preconditions are violated. Subsequently the planner can incorporate new facts and generate new solution plans.

Although these approaches deal with the same kinds of issues that we encounter in the AC voltage domain, the solutions proposed are domain specific and focused on the spatial reasoning involved in robot manipulation.

5.4 Models that Depend on External Solver

Our approach is similar to the semantic attachments in PDDL/M, but adopting the formalism described in Section 5.2, which explicitly distinguishes direct and indirect effects.

The variables of the problem are the predicate symbols and the numeric fluents in PDDL, whose domains are assumed to be $\{true, false\}$ and real numbers respectively. We restrict to the case of having as $V^{special}$ variables only numeric fluents. The actions are specified in the PDDL domain files, where direct effects can occur only to $V^{dep} \cup V^{indep}$ variables, while preconditions can be also expressed in terms of the $V^{special}$ variables. Whenever a change in at least one of the V^{dep} variables occurs, the state is also updated to calculate the set of $V^{special}$ variables according to the new values of the V^{dep} variables. We delegate the computation of the $V^{special}$ variables to an external solver. This allows the use of non-trivial algebraic operations.

Our approach can be considered a specialisation of Planning Modulo Theories (PMT) (Gregory et al., 2012), where arbitrary first order theories can be expressed and integrated into a planner as parameters.

To model indirect effects we extend the PDDL language to include *encapsulated types*, the implementation of which is hidden from the planner, but is provided by an external solver. The *encapsulated types* can have external evaluated functions, which correspond to the $V^{special}$ variables and to special functions that are interpreted by calling the external solver. The V^{dep} and V^{indep} variables are expressed using normal PDDL functions. We model the effects of the actions on the $V^{special}$ variables by assigning the results of an externally evaluated function to the object of the *encapsulated type*, which is interpreted by the external solver.

Similarly to PMT, we also provide an abstraction of the indirect effects, so that during the heuristic evaluation the indirect effects are computed by the abstraction.

Definition 22 (Abstraction of numeric indirect effects) *Given a state s, such that V^{dep} contains only numeric fluents, and a real number $M \geq 0$, a linear function is an* abstraction

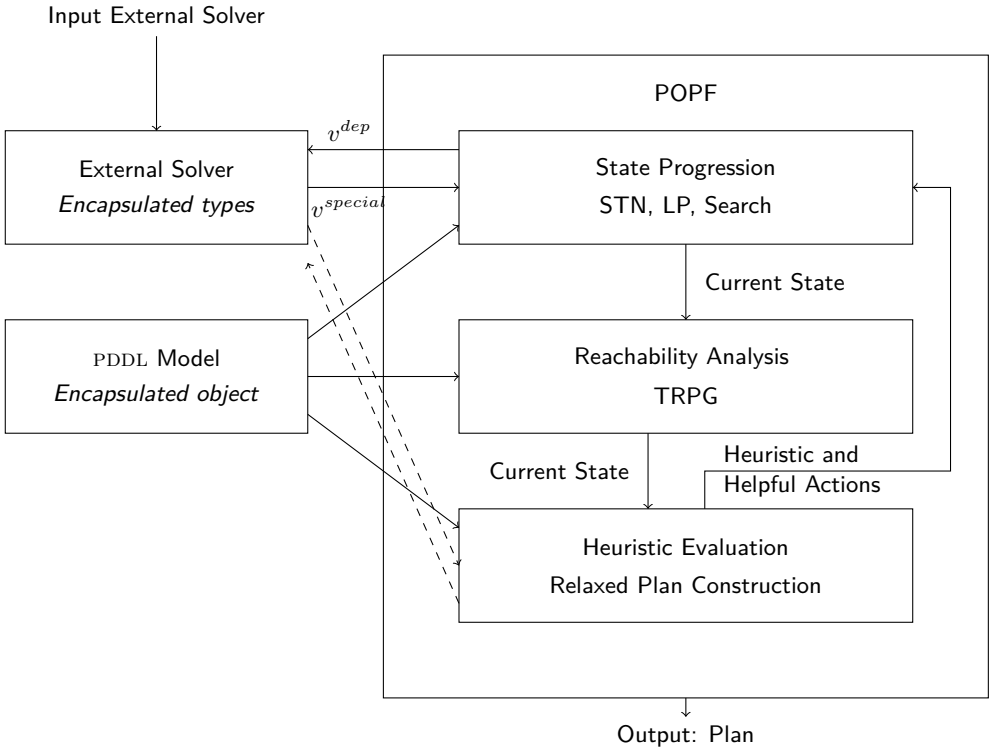


Figure 5.1: Planner and external solver implementation: the heuristic computation consults the external solver in computing TIF-grounded goals (see Definition 16) and, possibly, in computing action effects on the abstracted network representation (see Section 7.1.4).

of the indirect effects $\tilde{\phi}$, if for all valuation of V^{dep}

$$|\tilde{\phi}\left(v_1^{dep},...,v_n^{dep}\right) - \phi\left(v_1^{dep},...,v_n^{dep}\right)| \leq M.$$

In practice the abstraction of an indirect effect is an approximation of the effects of the $V^{special}$ that can be written as a linear function over the variables $V^{special}$ and V^{dep} and therefore it can be written in the PDDL language.

5.5 Implementation

In this section we describe the details of the implementation of the coupling between external solver and the planner to evaluate the indirect effects in Definition 20.

5.5.1 State Evaluation

The architecture of the interface of the planner POPF-TIF with the external procedures is shown in Figure 5.1. The planner takes as input the PDDL domain and the problem files and the external solver (the external module) associated with the PDDL domain. The PDDL

model contains all the variables, while their classification into the three categories $V^{special}$, V^{dep} and V^{indep} is contained in the interface to the external solver.

When the planner updates a state, as a first step the V^{dep} and V^{indep} are calculated. If there is a change in one of the V^{dep} variables, then the $V^{special}$ values are calculated calling the external solver, using as input all the values of the V^{dep} variables and other optional information independent of the transition function of the problem.

From a technical point of view, the external solver is implemented as a dynamically loaded shared library, that is passed as input to the planner when invoked. This allows the user to easily implement different external solvers. In order to successfully load the external solver, the planner and the external solver need to use a common interface. Because we take as input and output only numerical variables without any continuous effect, a state in the external solver is given by a map from the names of the fluents to their values. The interface consists of:

- a method that initialises the external solver, loading possible input parameters (such as the network structure in the case of the voltage control domain);
- a method that returns the list of V^{dep} variables;
- a method that returns the list of the $V^{special}$ variables;
- a method that calculates the updated $V^{special}$ variables from the V^{dep} variables;
- a method that calculates the abstraction of the indirect effects;

The dependency between the $V^{special}$ and V^{dep} variables is important for efficiency in the invocation of the external solver and for the correct updating of the state in the planner. Since we want to avoid calling the external solver when is not needed, only states with changed V^{dep} variables invoke the external solver. In addition, because the planner POPF-TIF is a partial ordering planner, it is more challenging to correctly update a state of the search space. As described in Section 3.7.3, a state in POPF-TIF (Definition 8) records also V^{eff}, the most recent steps that have instantaneous effects on the variables $v \in V$, and $VP(v)$, the set of step indices that depend on the variable v. When a change happens to a variable in V^{dep}, V^{eff} and VP must also be updated for the $V^{special}$ variables.

5.5.2 Abstraction of the Numeric Effects in Heuristic Evaluation

For the heuristic evaluation, the relaxation of numeric state variables used in POPF-TIF is based on the same principle introduced in metric-FF (Hoffmann, 2003), as described in Section 3.7.3, which can handle only numeric effects expressed in LNF. Since the effects on the $V^{special}$ variables cannot generally be expressed in LNF, we use the abstraction in Definition 22 to approximate the effect of the $V^{special}$ variables. The simplest effect that can be written is a linear effect of first degree:

$$v \mapsto v + c, \tag{5.1}$$

where c is a constant. This effect indicates whether an action increases ($c > 0$), decreases ($c < 0$) or is irrelevant ($c = 0$) to $v \in V^{special}$. The value of c should be chosen such that it

does not underestimate the real effect on v, so that we avoid excluding possible values in the reachability analysis. However, if the action can be applied multiple times, then the upper (or lower) bound of the reachable values becomes $+\infty$ (or $-\infty$) if the action increases (or decreases) the fluent value. In this case the actual value of c becomes less important during the building of the relaxed plan, but is still needed to decide the most appropriate action to choose during the extraction of the relaxed plan that is used to calculate the heuristic value.

Dynamic Update of the Numeric Effects

During the heuristic evaluation, the approximation of the effects are expressed as Linear Normal Form (LNF) effects, where the parameter c that appear in the formula is constant. Because in general the effects on the $V^{special}$ variables are non-linear, we add to the planner the option of automatically expressing the effects on the $V^{special}$ variables as in Equation (5.1), where the constant c is dynamically updated during the search process. Ideally, the update should be done at every state evaluation that changes at least one of the V^{dep} variables, however this operation is computationally expensive, because it requires one external call for each applicable action. For this reason, when we dynamically update the effects on the $V^{special}$ variables during the relaxed plan construction we set the reachable values to $(-\infty, +\infty)$ and only during the extraction of the relaxed plan, when actions with effects on the $V^{special}$ variables are required, over the parameters of the approximation recalculated, so that the most appropriate action can be inserted into the relaxed plan.

Interaction between Semantic Attachments and Timed Initial Fluents

It should be noted that if TIFs modify V^{dep} variables, then the external solver is called to compute the effects on the $V^{special}$ variables. When we use a *lookahead* heuristic we need to calculate the TIF-grounded goals $P[v/\alpha]$ (Definition 16). This implies that we need to call the external solver in the heuristic evaluation to determine the actual values of the $V^{special}$ variables that are used in $P[v/\alpha]$.

Chapter 6

PDDL Model of the AC Voltage Control Problem

In this chapter we describe the planning model developed for the AC voltage control problem using the PDDL language. For convenience, we refer to our modelling language as PDDL2.2, although it is actually PDDL2.2 extended with TIFs and the use of the encapsulated type.

The AC voltage control problem consists of manipulating network components, such as transformers, in time to interact with background profiles in such a way that the voltages at the busbars of the network always remain within their specified safety bounds, given background load and generation profiles over the period.

First we describe the domain, which contains the information about the types of objects and the action schemas and we explain how we use the *encapsulated type* to represent the *network*.

We then describe the problem file, which contains the information of the actual objects, the initial state and the goal condition of the single problem instances and we explain how we generate the problem files given an initial network and a series of timed data representing the profiles.

Finally, we describe the abstraction that we use to represent the effects of actions on the voltages during the heuristic evaluation.

6.1 PDDL2.2 Domain

Each component of the distribution system is represented as an object of a different *type*. In particular, the model specifies the types of Network, Busbar, Load, Generator and Device. The Device type breaks down into two subtypes: Tap and Capacitor. Since capacitors and reactors are both on/off devices, the Capacitor type is used for both. The Network type is encapsulated and there is only one object of this type, called *theNetwork*.

The problem is defined in terms of controllable and uncontrollable variables. The load and generation real and reactive power values are uncontrollable, while the transformer tap-levels and capacitor status settings (on/off) are controllable. Moreover, we need to classify the numeric variables into the three categories $V^{special}$, V^{dep} and V^{indep} for the

```
(:durative-action constraint-check
   :parameters ()
   :duration (<= ?duration (time-horizon)
   :condition (and (at start (S)) (at end (F))
      (over all (forall (?b - Bus) (and
         (<= (theNetwork.voltage ?b) (maximum-voltage ?b))
         (>= (theNetwork.voltage ?b) (minimum-voltage ?b))))))
   :effect (and (at end (is-end)))))
```

Figure 6.1: The `constraint-check` action in PDDL.

use of the external solver and associate the $V^{special}$ variable to the network type. In this domain the only $V^{special}$ variables are the voltages at each of the busbars of the network, which depend on the active and the reactive power of generators and loads, the transformer tap-level and the status of the capacitors. These are the V^{dep} variables, while all the other numeric quantities, such as the minimum and the maximum voltage limits, the time constants indicating the duration of the variables, etc. are classified as V^{indep} variables.

6.1.1 Model of the Voltage Constraints

The trajectory constraints involving the busbar voltages are modelled in PDDL2.2 with the action shown in Figure 6.1, following the compilation proposed in Section 4.2. The propositions `(S)` and `(F)` are the dummy propositions necessary to enforce the application of this action between the start and end of plan. In our model we want to model a trajectory constraint of the type `(hold-during` t_1 t_2 `P)`, thus we do not need to add a proposition p as precondition of every other actions, as in the case of the translation of the *always* constraint in Figure 4.4.

6.1.2 Model of the Actions

The available actions that we can use to modify the voltages are stepping up and down the taps of transformers and switching on and off capacitors. Once one of these actions is activated, it takes a time of the order of minutes to reach the desired effect, due to the mechanical constraints that govern the actual performance of the action. In addition, to avoid frequent operations, we impose a slack time period of the order of hours before the same device can be switched in the opposite direction. Each action defines three time-points: the moment in which the device receives the signal to perform the action, the moment in which the device action is concluded and we have the numeric effect, and the moment from which the device is allowed to perform an action in the opposite direction. Figure 6.2 shows a schematic representation of these operations. By contrast with other modelling languages, such as the one supported by the planner IxTeT (Laborie and Ghallab, 1995), in the PDDL2.2 language actions are defined only by two time-points, a start and an end, and intermediate effects are not allowed. If we want to model these operations in one single durative action, we can observe that the first and the second time-points are separated by a time interval much smaller then the interval between the second and the third time-points. Therefore, we can simplify the action collapsing the first two time-points into one. In Figure 6.3 we can see an example of this model for the stepping up of a transformer tap.

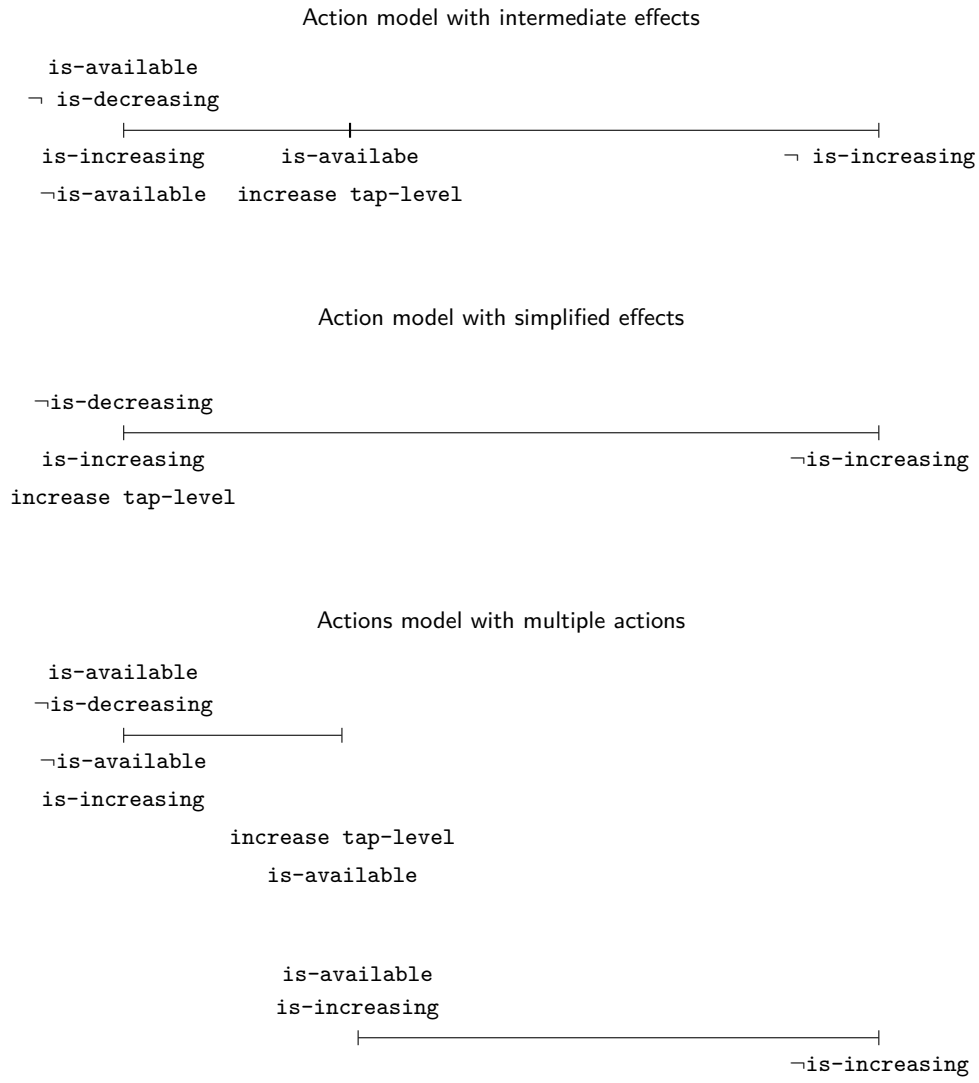


Figure 6.2: Time-points defined by the increase operation of the tap level of a transformer

On the other hand, if we do not want to lose the information on the delay of the effect, we need to combine two temporal actions at the cost of a bigger search space, as investigated in (Fox et al., 2004). The model of these two actions for the step-up operation of transformer taps is shown in Figure 6.4. The `step-up` action is modelling the delayed effect on the network, while the `release-increasing` action ensures that at least a slack time period is passed between two subsequent step-change operations in the opposite direction. The `step-down` and `release-decreasing` actions can be written analogously. We decided to adopt this second model of actions because the time to perform mechanical tap changes is significant compared with the time over which demand can vary.

In Figure 6.5, the actions for capacitor and reactor activation are presented. The effect of these actions is the change of the status of the capacitor from active to inactive. Analogous actions are available for the deactivation of the capacitors (or reactors).

We now have the elements of a BTMP, because the voltage is a function of both the

```
(:durative-action simple-step-up-tap
   :parameters (?t - tap)
   :duration (= ?duration (slack-period ?t - tap)
   :condition (and (at start (is-not-decreasing ?t)))
   :effect (and (at start (increase (tap-level ?t) 1))
      (at start (not (is-not-increasing ?t))) (at end (is-not-increasing ?t))))
      (at start (increase (theNetwork) (step-up-effect ?t)))))
```

Figure 6.3: `step-up-tap` action in PDDL.

```
(:durative-action step-up-tap
   :parameters (?t - tap)
   :duration (= ?duration 0.1)
   :condition (and
      (at start (is-not-decreasing ?t))
      (at start (< (tap-level ?t) (max-tap-level ?t)))
      (at start (is-available ?t)))
   :effect (and
      (at start (not (is-not-increasing ?t)))
      (at end (is-increasing ?t))
      (at start (not (is-available ?t)))
      (at end (is-available ?t))
      (at end (increase (tap-level ?t) 1))
      (at end (increase (theNetwork) (step-up-effect ?t)))))

(:durative-action release-increasing
   :parameters (?t - tap)
   :duration (= ?duration (slack-period ?t))
   :condition (and (at start (is-increasing ?t)))
   :effect (and (at end (is-not-increasing ?t))
      (at start (not (is-increasing ?t)))))
```

Figure 6.4: The `step-up-tap` and `release-increasing` actions in PDDL.

controllable and uncontrollable metric fluents, and hence is a mixed metric expression. All of the metric fluents in the problem are controllable or uncontrollable. We have a *flexible* trajectory constraint as specified in Definition 14, therefore the AC voltage control problem is a BTMP.

As shown in Equations 2.2, the function determining the linkage between system voltage and the component variables in the AC case is very complex. In particular, the tap-levels and capacitor-settings determine the values of reactance and susceptance on the lines on which those components appear. Voltage is also a function of the precise network topology defining how the components are linked. The external solver computes the voltages at different components of the network whenever it is invoked by the planner. The way in which the planner notifies *theNetwork* that it has changed state, as a result of the application of an action, is by means of the effect `(increase (theNetwork) (`δ`))`, where δ is a function of the affected tap or capacitor. The external solver is responsible for computing the new network state so that the planner has access to the updated busbar voltages.

We note that the keywords *increase* and *decrease* are being overloaded to refer to an encapsulated object, and the precise meaning of the expression `(increase (theNetwork) (`δ`))` depends on the implementation of the Network type. The implementations of the functions `(step-up-effect ?t - tap)`, `(step-down-effect ?t - tap)` and `(capacitor-effect`

```
(:durative-action release-capacitor
   :parameters (?t - capacitor)
   :duration (= ?duration 3)
   :condition (and (at start (is-available ?t)))
   :effect (and (at end (is-available ?t))
      (at start (not (is-available?t)))))

(:durative-action activate-capacitor
   :parameters (?c - capacitor)
   :duration (= ?duration 0.1)
   :condition (and
      (at start (is-not-active ?c))
      (at start (is-available ?c)))
   :effect (and
   (at start (not (is-not-active ?c)))
   (at start (not (is-available ?c)))
   (at end (is-active ?c))
   (at end (increase (theNetwork) (capacitor-effect ?t))))
```

Figure 6.5: The `activate-capacitor` action in PDDL.

`?c - capacitor`) are provided by the external solver, and the signatures of these functions are part of the interface between the planner and the solver. Thus, the planner knows that it is applying an action that changes the network state, but does not have access to a declarative description of the effects of those changes on the network. An important advantage of the encapsulation is that we can define an abstraction of the Network type for the planner to use during the heuristic evaluation of a state. The specific abstraction provided by our implementation is discussed in Section 6.3.

6.2 PDDL2.2 Problem

The problem file contains the information about the single problem instance. All the objects are instantiated, the initial state is specified and the goal conditions are stated. In the AC voltage control problem, the goal is simply to reach the end of the time horizon without violating the voltage constraint, which is ensured by the `constraint-check` action. The goal condition is then expressed as:

```
(:goal (is-end))
```

where the proposition `is-end` is only added at the end of the `constraint-check` action.

In the initial state, we need to specify the initial values of all the numeric fluents and the constant parameters of the actions, such as the minimum and maximum voltage that each busbar can have. The `(S)` and `(F)` propositions are also added and deleted to force the application of the `constraint-check` action. In addition, we need to specify the prescheduled values of the generators and load profiles, which are the uncontrollable events influencing the voltages, expressed as TIFs. For each time-point and each load/generator we need to specify the active and the reactive power absorbed/injected. Typically, for a medium size network containing 10 loads and 5 generators and considering a time horizon of 24 hours, with time-step of 30 minutes, we need about 1440 TIFs to express the changing profiles.

6.2.1 Generation of the Problem Files

The generation of the problem file is done automatically, given the network topology, the load and generation profiles, and the constant parameters of the problem.

The network configuration is expressed as a MatPower file (a standard file system format used by power systems software tools (Zimmerman et al., 2011)), which is also an input file of the external solver linked to the planner. From this file we extract the information on busbars, loads, generators, capacitors and transformers. In particular we need a unique name for each network component that relates the PDDL objects to the MatPower elements, and the initial values of the network. The loads and generations profiles are given as lists of timed values. The values are scaled by constant factors in order to maintain the relative magnitudes of loads and generators. Loads and generators are assumed to operate at a constant *power factor*, meaning that the ratio between active and reactive power is constant.

6.3 Abstraction of the Network Effects

The reachable values of the $V^{special}$ variables are not accessible to the planner, so, in order to obtain any heuristic information from the external solver, the planner requires a relaxation of the Network encapsulated type. In this relaxation is a simplification of the effects of tap-changing and capacitor-switching actions on the voltages at the busbars. We relax these effects into simple linear first order effects, according to the Definition 22 in Section 5.4. We treat as zero the effects on any busbars that are far away from the transformer or capacitor that is acted upon. This relaxation is sufficiently simple to express that it can be declaratively described in PDDL2.2. We therefore provide a collection of constants in the initial state, one for each capacitor, tap and busbar, of the form `(step-min ?d - device ?b - busbar)` and `(step-max ?d - device ?b - busbar)`, to represent the relaxed voltage updates, and we translate the `increase` and `decrease (theNetwork) (`δ`)` expressions into quantified simple effects of the tap-changing and capacitor-switching actions. As an example, for heuristic evaluation, the `(step-up-tap ?t - tap)` action contains the quantified effect:

```
forall (?b - bus) (and
          (at start (increase (theNetwork.voltage ?b)
                              (step-max ?t ?b))))
```

The values of the `(step-max ?t ?b)` are provided by the external solver, and they can be static, specified in the initial state or uploaded dynamically as described in Section 5.5.2. These values are obtained by applying the action to the initial state and calculating the differences between the previous and the new voltages of the busbars.

6.4 Overview of Solvers for the Voltage Control Problem

For the voltage control problem, the external solver needs essentially to calculate the power flow equations as presented in Equations 2.2. In order to execute the calculation, the external solver needs as input the configuration of the network (the links between the different elements) and information about the resistance and impedance of wires. An advantage

Network	ρ_δ				ρ_V			
	IPSA	LPAC	DC	NR	IPSA	LPAC	DC	NR
ieee9	1.000	1.000	0.999	1.000	1.000	0.999	0.000	1.000
ieee14	1.000	1.000	0.999	1.000	0.942	0.936	0.597	1.000
ieee30	1.000	0.999	0.980	1.000	1.000	1.000	0.000	1.000
ieee39	1.000	0.999	0.995	1.000	1.000	0.999	0.444	1.000
ieee57	0.999	0.999	0.990	1.000	0.963	0.958	0.277	1.000
ieee118	1.000	0.999	0.937	1.000	0.994	0.994	0.691	0.999
ieee300	0.941	0.998	0.978	1.000	0.815	0.950	0.562	1.000

Table 6.1: Correlation of the phase angle and the voltages at the busbars calculated by different methods and compared with MatPower.

of using an external solver, instead of trying to model everything in PDDL, is that we do not need to express variables in the domain that are not relevant to the search (such as impedance), but are necessary to perform the power flow calculations.

We considered several solvers, including:

- IPSA (TNEI Services Ltd, 2010): this is a power engineering licensed software specialising in AC power flow calculations. The computational cost of communicating with IPSA realised with a Python interface, is too high, so the use of this solver is practically infeasible.
- MatPower solver (Zimmerman et al., 2011): this is a free Matlab package for solving power flow equations and optimal power flow. Also in this case, the computational cost of communication limits the use of this software as a external solver of the planner, however we use it as validation tool to compare the results of the other solvers.
- DC approximation of the power flow equations: although the DC approximation, as presented in Equation 2.3, is a popular approximation in the power engineering literature, this approximation makes the assumption that the voltage is always constant, and therefore this method cannot be use to control the voltage.
- LPAC approximation of the power flow equations (Coffrin and Van Hentenryck, 2012): the AC power flow equations are approximated as a LP problem, providing a piece-wise linear approximation of the cosine. Although this approximation cannot be directly used to make decisions on transformers, due to the non-linearity of the transformer term, this method can be considered as alternative to the full AC power flow equations. We have experimented with an implementation in which we approximated the cosine in 20 linear pieces. The low accuracy of the model and the high computational time, due to the presence of a LP problem, make this approach not suitable for our purpose.
- Newton-Raphson method (Ypma, 1995) for the AC power flow equations: this is an iterative numeric algorithm that allows us to find successively better approximations to the zeros of a function.

All these solvers are implemented and compared in terms of computation time and in terms of accuracy with respect to the MatPower solver.

To measure the accuracy of the solver we use the correlation of the voltage and the phase angles of the results obtained with MatPower and another solver. The results are shown in

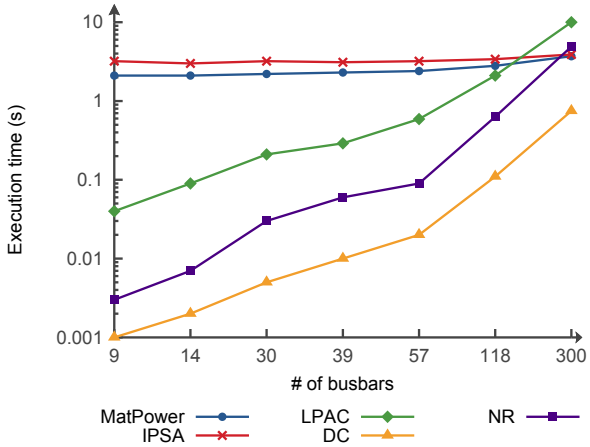


Figure 6.6: Execution times of different IEEE benchmark problems for different external solvers.

Table 6.1, while the computational time of the IEEE benchmarks (University of Washington Electrical Engineering, 2015) for each external solver is shown in Figure 6.6. Among all the possibilities, the fastest method is always the solution using the DC approximation of the power flow equations. This method, however, neglects the variation of the voltage of busbars, thus, it cannot be used for the voltage control problem. The best solver in terms of execution time and accuracy is the Newton-Raphson method which outperforms all the others with networks up to 300 busbars. This is due to the communication cost of calling an external software such as IPSA or MatPower. For this reason we choose as external solver our own implementation of the Newton-Raphson algorithm.

Chapter 7

Experimental Results

In this chapter we present the results obtained with the planner POPF-TIF. The chapter is divided into two distinct parts. In the first part we present the results for the AC voltage control problem, presenting the case study of the AuRA-NMS network presented in Section 2.2.3. We compare the plans obtained with our model and planner with the current *state of the art* reactive approach used in distribution networks. The results show that, in case of meshed network, the planning approach finds solution with better qualities, both in terms of smaller violations and fewer number of actions. We then analyse the scalability of our approach, showing that the lookahead heuristic allows the planner to solve more problems than when using the TRPG heuristic. The second part of the chapter is dedicated to a comparative study of the *lookahead* heuristic and *mixed search strategies* for various BTMPs. Both the lookahead heuristic and the mixed search strategies contribute to increase the number of problems solved and to decrease the execution time of the planner POPF-TIF. POPF-TIF is the only planner that we are aware of that can solve large instances of BTMPs.

7.1 Voltage Control Problem

In this section we present and discuss results of the application of POPF-TIF to the AC voltage control domain. First we present an illustrative example where our planner finds a sequence of actions to manage AC voltage control in the 33kV network of Section 2.2.3. Our control strategy is compared with a modern reactive control approach, where transformers are reactively controlled and co-ordinated through the use of sensors and time-delays. We then examine the scalability of the planner in terms of the size of the network, showing that the integrated architecture supports good scaling behaviour. Although, in principle, we can reduce the size of a zone in order to make it amenable to planning, larger zones offer lower variance in the predicted load and generation profiles as the number of generators and consumers attached to the zone increases. It is therefore an important benefit to be able to scale the performance of the planner to larger networks. To demonstrate the robustness of plans in the presence of uncertainty in the predicted load and generation profiles, we perform experiments in simulation which are reported below. Finally we present a brief study of the accuracy of the first order approximation of the effects of actions on the $V^{special}$ variables in

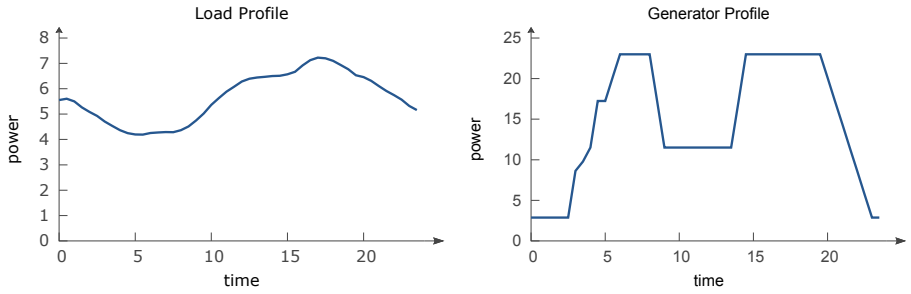


Figure 7.1: Typical load and generation profiles.

the heuristic evaluation. All the experiments are performed on a 3.4 GHz Intel Core i7-2600 machine, limited to 20 minutes and 4 GB of memory. The 20 minute limit reflects the fact that, given the half-hour frequency of changes in the predicted load and generation profiles, decision-making can only be responsive if planning can be done within this granularity.

7.1.1 Case Study: a 33 kV network

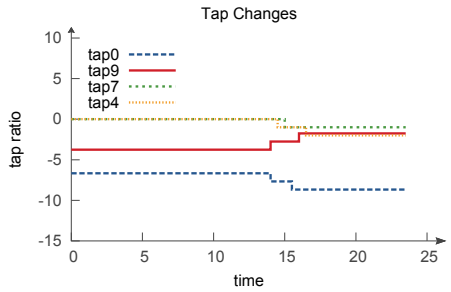
As a case study we use the 33-kV interconnected AuRA-NMS network described in Section 2.2.3. We consider the AC voltage control problem over the period of 24 hours, given that the load and generation profiles are described at 30 minute intervals. All generators, other than Gen1, are assumed to be Combined Heat and Power Plants with different capacities, but with deterministic output. Figure 7.1 shows typical load and generation profiles.

The output plan generated by POPF-TIF, using the domain described in Section 6.1 and the problem instance with load and generation profiles of Figure 7.1 is the following:

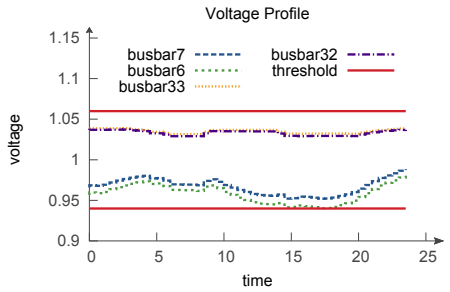
```
0.000: (constraint-check)  [24.001]
13.901: (step-down-tap tap0)  [0.100]
13.902: (step-up-tap tap9)  [0.100]
14.401: (step-down-tap tap4)  [0.100]
14.901: (step-down-tap tap7)  [0.100]
15.401: (step-down-tap tap0)  [0.100]
15.901: (step-up-tap tap9)  [0.100]
16.401: (step-down-tap tap4) [0.100]
```

where each line indicates the time at which the action starts, the name of the action, and, in square brackets, the duration of the action. In this simple example, few control actions are required and there is no oscillation of tap-settings. Therefore, the slack-period does not affect the timings of the actions. The actions on different taps overlap where possible, in order to interact successfully with the TIFs. It should be noted that the first action is the start of the envelope action that ensures that the constraints are met. The unit of time corresponds to one hour.

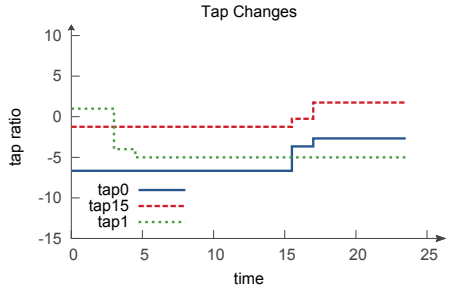
POPF-TIF with 1 lookahead solves this problem by searching 66 states in 0.59 seconds. In contrast, the same problem is solved, with a different plan, by POPF-TIF with 0 lookahead after searching 7610 states in 15.3 seconds. This shows the benefit obtained from the information provided by the lookahead.



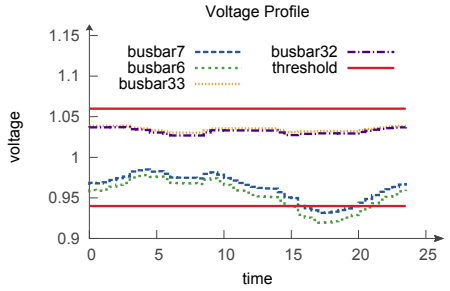
(a) Tap ratio profiles of selected transformers obtained by the planner.



(b) Minimum and maximum voltage profiles obtained by the planner.



(c) Tap ratio profiles of selected transformers obtained by the HTC method.



(d) Minimum and maximum voltage profiles obtained by the HTC method.

Figure 7.2: Results comparing POPF-TIF with the HTC method in solving the same problem instance, provided in the online appendix

Figures 7.2a and 7.2b show the tap operations of the transformers 0, 4, 7 and 9 and the maximum and minimum voltage profiles of the relevant busbars in the network operating according to the above plan.

To show what happens when we require the same tap to be stepped up and down, and therefore need to release the tap before stepping it in the other direction, we present the following plan which is obtained using the same domain description and the small problem example in Appendix C.

```
0.000: (constraint-check)  [24.001]
0.000: (step-down-tap tap0)  [0.100]
0.101: (release-decreasing tap0)  [3.000]
9.901: (step-up-tap tap0)  [0.100]
14.901: (step-up-tap tap0)  [0.100]
15.002: (release-increasing tap0)  [3.000]
19.901: (step-down-tap tap0)  [0.100]
```

We see that, having stepped-down `tap 0` at time 0, we have to complete the `release-decreasing` action before being able to step the same tap up. We note that `tap 0` is actually not stepped up again until time 9.901, followed by another step-up at 14.901. The tap is prevented from being stepped down until 17.902, but is not actually stepped-down until 19.901. These delays are in response to the TIFs of the problem, which occur at times 5, 10 and 15. The stepping actions of the plan are timed to respond to these while satisfying the slack period constraints enforced by the `release-increasing` and `release-decreasing`

actions.

Our first experiment compares the POPF-TIF approach with the Hierarchical Tap Changing Method (see Section 2.2.2). In the AuRA-NMS network, the hierarchical structure used is as follows:

(i) Tap 5, Tap 6, Tap 8, Tap 14 are transformers that perform voltage step down from the grid to the distribution network and provide regulation of the voltage at this level. These are at the top of the hierarchy.

(ii) Tap 1, Tap 0 and Tap 11 are the transformers that control the load variation when distributed generation is inactive or providing low output. These taps operate after the taps in the first group.

(iii) Tap 15 regulates the voltage due to load variations.

(iv) The remaining transformers are the last ones to act, being the ones responsible for ensuring that distributed generators are safely connected to the rest of the network.

Using the HTC approach, when the direction of power flow opposes the normal flow from the main generation sources, voltages on the upstream busbars might fall outside their safety bounds. An example of such a case is shown in Figures 7.2c and 7.2d, where the HTC approach is applied to our example. As we can see from the plots, the HTC approach violates some voltage constraints, while the planning approach finds a solution that does not violate any constraint and changes a smaller number of taps. The above observation demonstrates the advantage of the planning approach over the HTC approach in meshed networks with distributed generation.

The performance of the planning approach depends on the accuracy of the load-series predictions. We now explore the extent to which this is the case.

The assumption that we can obtain adequately accurate predictions is valid in current networks for which we have a large amount of historical data. It is, however, challenged by the expected structure of future networks. There will be more ad hoc generation and more load variability than in current networks. In particular, zoning will increase variance by reducing the number of consumers connected to the supply sources in any zone.

For this reason we test the robustness of the generated plan, starting from a known load profile and applying the same plan to the network with random variations of the load profile. The different profiles are obtained by scaling all the loads at each time-point using a random number that follows a normal distribution with parameters $\mu = 1$, $\sigma^2 = 0.001$, as shown in Figure 7.3. The comparison with the HTC approach, which is run for each new load profile generated, is done by measuring the sum of total violations of voltage for each busbar in the network. This experiment is performed by generating 500 variations of each of 12 different load profiles. The total violation $\mathcal{V}_{tot}$ of a plan with respect to a load profile is measured as the sum of the violations of voltage level at each busbar:

$$\mathcal{V}_{tot} = \sum_{t \in times} \sum_{k \in busbar} \mathcal{V}_{tk}, \tag{7.1}$$

where *times* is the set of time-points in the execution trace resulting from applying the plan,

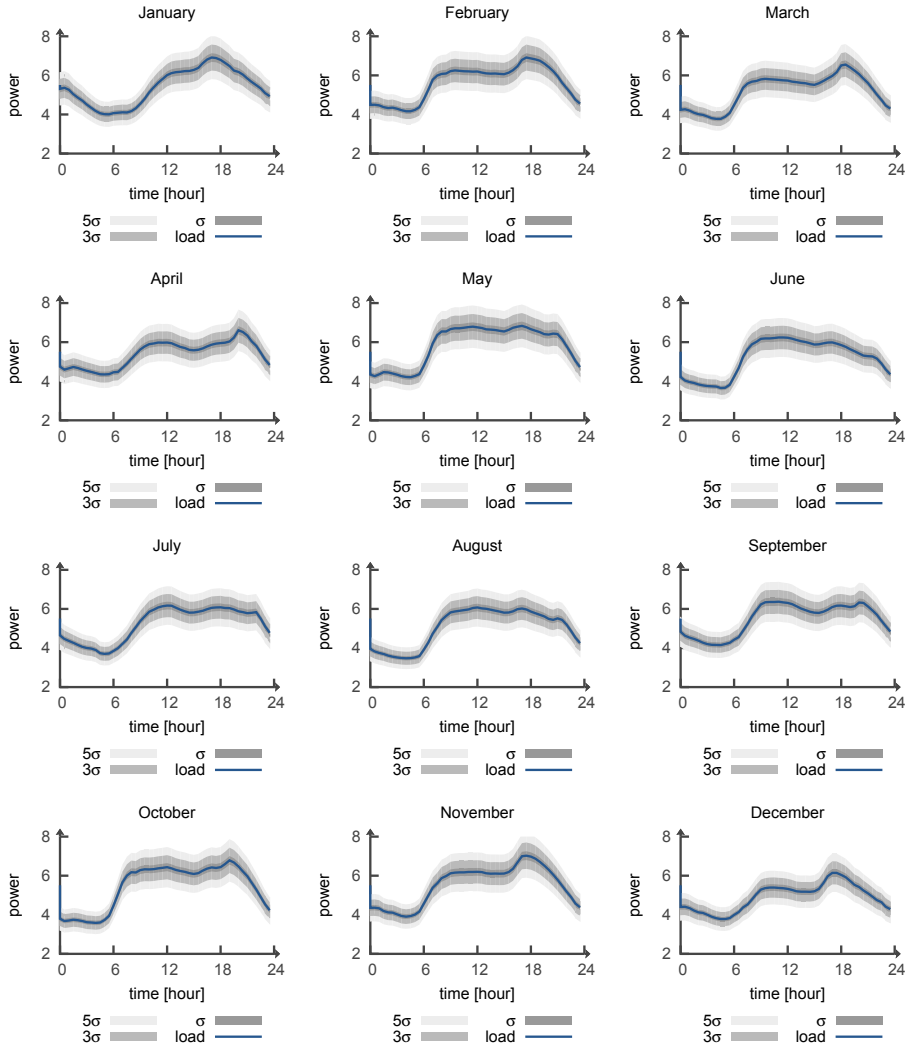


Figure 7.3: Variation of the 12 load profiles chosen as representative of a year.

or HTC approach, over the 24-hour period, and:

$$\mathcal{V}_{tk} = \begin{cases} 0 & \text{if } V_{k,min} \leq V_{t,k} \leq V_{k,max} \\ V_{t,k} - V_{k,max} & \text{if } V_{k,max} \leq V_{t,k} \\ V_{k,min} - V_{t,k} & \text{if } V_{t,k} \leq V_{k,min} \end{cases} \tag{7.2}$$

The quantity $\mathcal{V}_{tk}$ is the total violation at time t and busbar k. The quantity $V_{t,k}$ is the voltage at time t at busbar k. As is standard in power systems, all voltage measurements are given in per unit volts, meaning that the voltages at a busbar are scaled relative to the nominal target voltage at that busbar. The comparison between the total violations incurred by the planning approach and the reactive approach is shown in Table 7.1, where the columns represent the total violation of the original load profile, the mean and standard deviation of the total violation and of the number of tap changes operated for the different variations of the load profiles.

In the planning approach the same plan is applied to the set of 500 instances for each

Load profile	Planning					Reactive				
	$\mathcal{V}_{tot}$	$\mu_{\mathcal{V}_{tot}}$	$\sigma_{\mathcal{V}_{tot}}$	$\mu_{\#tap}$	$\sigma_{\#tap}$	$\mathcal{V}_{tot}$	$\mu_{\mathcal{V}_{tot}}$	$\sigma_{\mathcal{V}_{tot}}$	$\mu_{\#tap}$	$\sigma_{\#tap}$
profile 1	0.000	0.005	0.003	7.0	0.0	0.193	0.202	0.024	13.0	1.7
profile 2	0.000	0.007	0.004	12.0	0.0	0.697	0.718	0.040	25.0	1.0
profile 3	0.000	0.004	0.003	12.0	0.0	0.549	0.551	0.032	24.0	1.7
profile 4	0.000	0.004	0.003	3.0	0.0	0.032	0.043	0.022	8.0	1.0
profile 5	0.000	0.004	0.003	7.0	0.0	0.347	0.351	0.038	12.0	2.4
profile 6	0.000	0.004	0.003	1.0	0.0	0.000	0.039	0.027	10.0	1.4
profile 7	0.000	0.003	0.003	2.0	0.0	0.000	0.027	0.025	9.0	1.4
profile 8	0.000	0.004	0.003	1.0	0.0	0.040	0.048	0.020	11.0	1.4
profile 9	0.000	0.007	0.004	2.0	0.0	0.054	0.045	0.027	9.0	1.0
profile 10	0.000	0.005	0.003	6.0	0.0	0.232	0.253	0.039	12.0	1.0
profile 11	0.000	0.007	0.004	13.0	0.0	0.616	0.618	0.039	25.0	1.0
profile 12	0.000	0.009	0.005	11.0	0.0	0.410	0.425	0.040	23.0	2.0

Table 7.1: Total violation of different load profiles. The first column, $\mathcal{V}_{tot}$, for each of the two approaches, shows the total violation when a trajectory corresponds precisely to the predicted load profile. The $\mu_{\mathcal{V}_{tot}}$ and $\sigma_{\mathcal{V}_{tot}}$ columns show the mean and standard deviation of the violation when the load profile varies according to the distribution described previously. The $\mu_{\#tap}$ and $\sigma_{\#tap}$ are the mean and standard deviation of the number of tap changes initiated by each of the approaches over all of the loads.

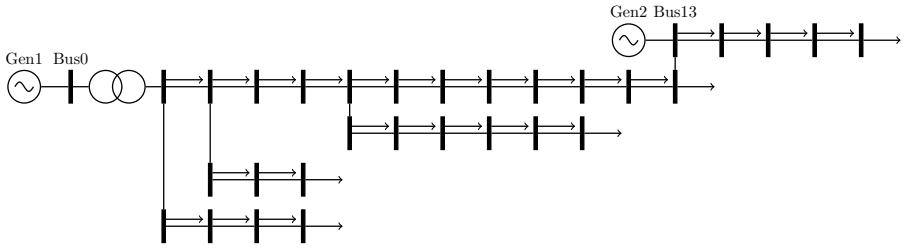
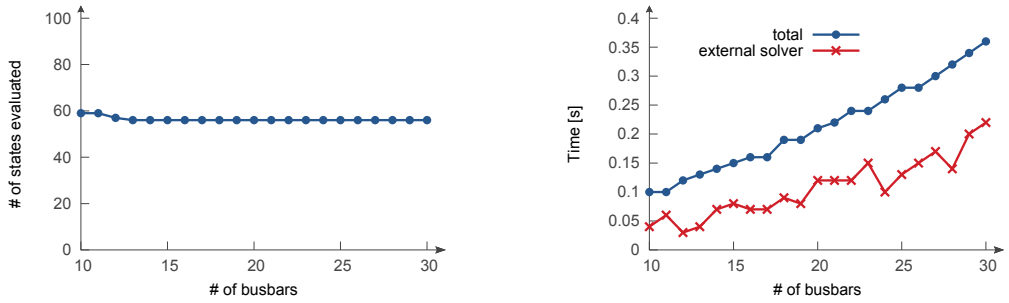


Figure 7.4: A four-feeder distribution network (Prenc et al., 2013)

of the 12 load profiles. This is why the number of tap changes remains constant and the standard deviation is zero. As shown in the table, the planning approach never violates the constraints when there is perfect knowledge of the load-series. However, when there is imperfect prediction, there are cases where it fails to stay within the specified bounds. The HTC approach always performs more tap changes than the planning approach and, despite not relying on any forecast, there are cases in which voltage limits are violated on some of the busbars. Within the range of profiles examined, the total violations incurred by the planning approach are always lower than those incurred by the HTC approach. The above results indicate the need for carefully decomposed zones in conjunction with improved forecasting techniques and show the merit of developing more sophisticated reasoning techniques for solving the AC voltage control problem.

7.1.2 Scalability with respect to the Size of the Network

To test the scalability of our approach with respect to network size, we consider problems with networks with an increasing number of busbars and lines. We take as our test network the distribution network described in (Prenc et al., 2013). The original network, shown in Figure 7.4, consists of 4 feeders with a total of 30 busbars. Bus0 represents a medium voltage



(a) Number of States Evaluated as a function of the number of busbars in the network

(b) Execution time as a function of the number of busbars in the network

Figure 7.5: Scalability of planner performance as the number of busbars increases

busbar of a high/medium voltage (HV/MV) primary substation. An additional generation unit is connected to Bus13. A plan is required for a 24-hour period, given a load profile at half-hour granularity. We run the model with a load profile taken from the National Grid data set for a winter's day of 2012 (National Grid PLC, 2012). In order to see how our system scales in terms of the size of the network, we solve the problem with networks with a decreasing number of busbars. Results are shown in Figure 7.5. In the left plot, Figure 7.5a, the number of states evaluated as a function of the number of busbars (the complexity of the network) is shown and it indicates that, apart from smaller networks, the number of states evaluated during the search is constant. In the right plot, Figure 7.5b, the total execution time and the time consumed by the external solver are presented. In this plot we can see that the execution time increases slightly super-linearly as the network size increases. It can also be seen that the time taken by both the external solver and the planner are well within feasible bounds for a real-time solution approach. The planner time includes the external solver time and all communications between the planner and the solver.

7.1.3 Scalability with respect to the Number of Control Points

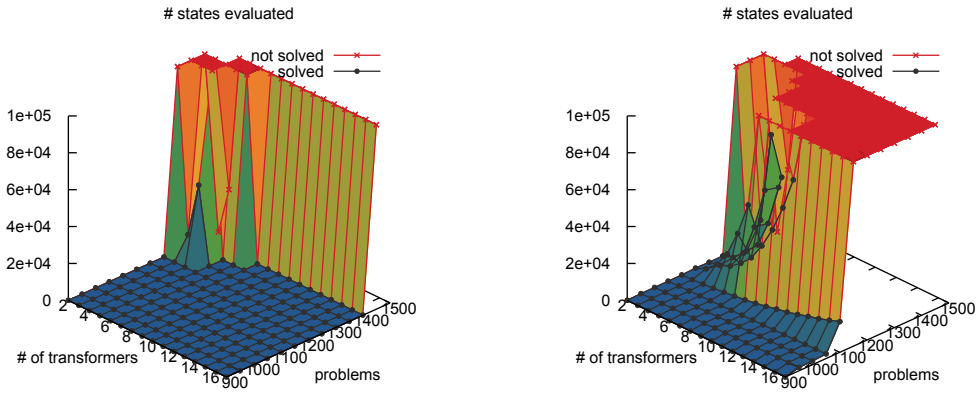
We now explore the scalability of POPF-TIF with respect to the number of control points present in the network. For this test we take the 33kV AuRA-NMS (Davidson et al., 2010) network shown in Figure 2.7.

For this domain we generated different problems changing the load profiles and the number of transformers present in the network. We start with two transformers, increasing incrementally until the configuration in Figure 2.7 is reached.

We run POPF-TIF using two different heuristic settings:

- a lookahead of 1 (1 lookahead);
- a lookahead of 0 (0 lookahead, corresponding to the standard heuristic setting of POPF2).

In Figure 7.6 we can see how the search scales with the number of control points. The dots represent the number of states explored during the search for solutions to problems with an increasing number of control points (transformers) and with increasing difficulty



(a) Results using POPF-TIF with 1 lookahead (b) Results using POPF-TIF with 0 lookahead

Figure 7.6: Number of states evaluated as a function of the number of control points and problem difficulty. The surface is colour-coded to indicate its height. The red points represent problems not solved while the black points represent problems solved.

measured in terms of the variability of the load profile. In the left plot (Figure 7.6a) 1 lookahead is used, while in the right plot (Figure 7.6b) 0 lookahead is used.

We also consider versions of these problems in which the changes of load and generation profiles are modelled using TILs and dummy actions, rather than TIFs, according to the compilation presented in Section 4.1.1. Figure 7.7 shows the results. In this case, the easiest problem is solved for all the networks, another one is solved for networks with up to 16 transformers and few others with networks with up to 8.

7.1.4 Evaluation of the Network Abstraction

In this section we test the effectiveness of the network abstraction in four different configurations of the planner. For this evaluation we start with the four-feeder distribution system, shown in Figure 7.8 and based on (Su and Lee, 2003). We then create a problem set by increasing difficulty along two dimensions. On one dimension we increase the number of capacitors. On the other, we increase the variability of the load profile. The problems using 5 and 6 capacitors are obtained by removing capacitors C1 and C3 from the network. The problems with more than 7 capacitors are obtained by successively adding capacitors to busbars B3, B4, B7, B8, B9, B10 and B15. All of the problems are subjected to the same 6 load variations. This gives us a collection of 60 problems in total. The purpose of the experiment is to investigate the trade-off between the informativeness of the heuristic and the time spent computing it. The four configurations considered are the following:

(a) 0 lookahead and no use of the network abstraction (i.e: `(step-max ?c ?b)` = `(step-min ?c ?b)` = 0, for all capacitors and busbars);

(b) 1 lookahead and a simple network abstraction in which capacitor effects are treated as an identical constant for all capacitors that have a non-negligible effect on a busbar, 0 otherwise (neighbourhood abstraction);

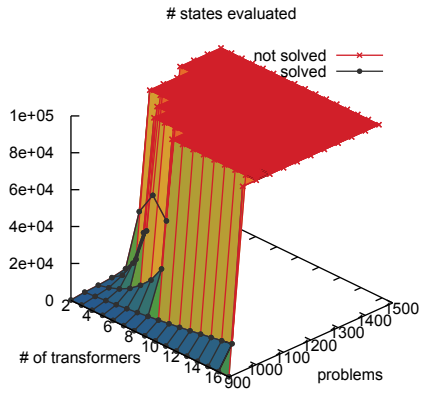


Figure 7.7: Number of states evaluated as a function of the number of control points and for problems with increasing difficulty. The red points represent unsolved problems and black points are solved problems.

(c) 1 lookahead and a network abstraction defined in the initial state (static first order approximation);

(d) 1 lookahead and state-dependent update of the network abstraction. That is, the capacitor effects are treated as $V^{special}$ variables and updated in each state by a call to the external solver (dynamic first order approximation).

In case (a) the heuristic contains no information about the effects of actions or TIFs on the voltage. It is not necessary to consider 0 lookahead in any of the other cases, since the network abstraction would be ignored if 0 lookahead were used.

The results are shown in Figure 7.9. The plots show the number of states evaluated in the four different configurations of the heuristic for the different load profiles and for problems with an increasing number of capacitors. The number of transformers and generators is fixed throughout. As we can see, configurations (a) and (b) lead to many states being explored. The approximation of the indirect effects is highly accurate for both configurations (c) and (d). The difference between these two configurations in terms of states explored is minimal. Configuration (d) searches slightly fewer states in the middle difficulty problems, although this cannot be seen in the figure. It searches more states in the hardest problems, which is an artefact of the tie-breaking between equal-valued states in the search for a plan.

Figure 7.10 shows the CPU time required by the four configurations of POPF-TIF on the 6 problem sets corresponding to each of the 6 load profiles. Configuration (a) is very expensive in all cases, and cannot solve the harder problems. In problem sets 1, 2, 3 and 4, the neighbourhood-based heuristic, configuration (b), is just as good as a more accurate approximation. However, in problem sets 5 and 6 this ceases to be the case. Configuration (c) performs well in all of the problems, while configuration (d) becomes too expensive as the problem difficulty and number of capacitors grow.

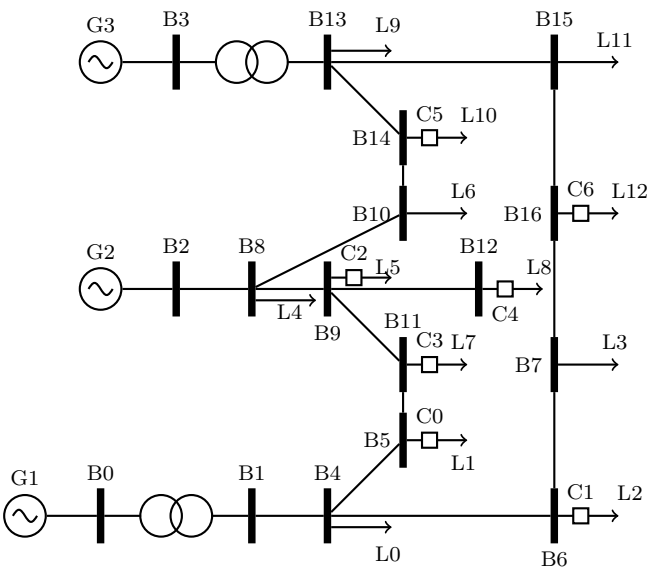


Figure 7.8: Four-feeder distribution network (Su and Lee, 2003).

7.2 Evaluation of Lookahead

In this part of this chapter we evaluate the *lookahead* heuristic and the two *mixed search strategies* in comparison with the standard TRPG heuristic and search in POPF-TIF. This study is conducted by taking into account a variety of BTMPs.

All tests in this case use a 3.4 GHz Intel Core i7-2600 machine, limited to 30 minutes and 4 GB of memory.

7.2.1 Benchmark

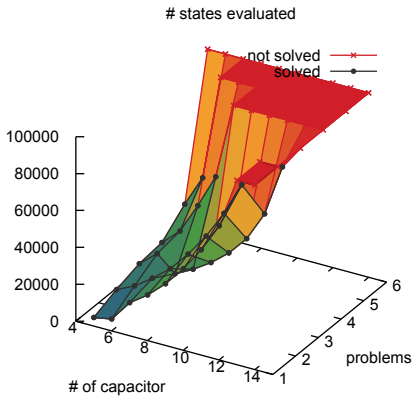
Since the class of BTMPs has not been widely explored in the literature on planning with PDDL, there are no benchmark domains available. We therefore introduce a variety of domains that capture key features of the class. Our total benchmark consists of 7 domains with 30 problems each, for a total of 210 problems. The domain descriptions of the BTMPs considered are presented in the following.

AC Voltage Domain

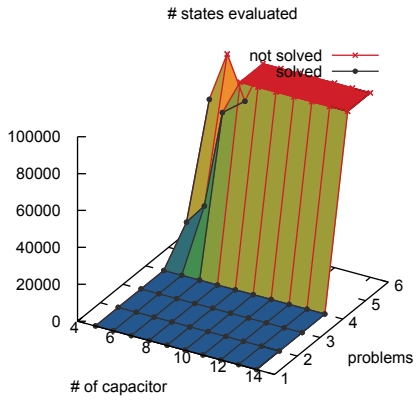
This is the domain described in Section 6.1. Although results of this domain are presented in the previous section, we include this domain to evaluate the effect of the *multiple lookahead* heuristics and the *mixed search strategies*.

Unit Commitment Problem

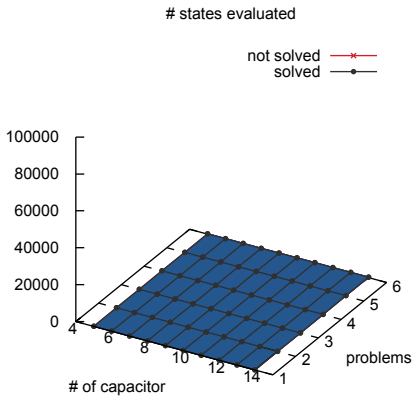
The Unit Commitment Problem (UCP) is the problem of deciding which generation units in a power system to switch on at what time. A closely related problem is the *Economic Dispatch (ED)*, which is the problem of deciding the output levels of each unit that are currently



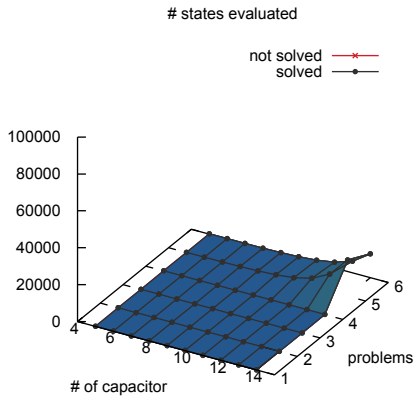
(a) No network abstraction



(b) Neighbourhood abstraction



(c) Static first order approximation



(d) Dynamic first order approximation

Figure 7.9: Results of using the four-feeder distribution system and the four configurations of the heuristic.

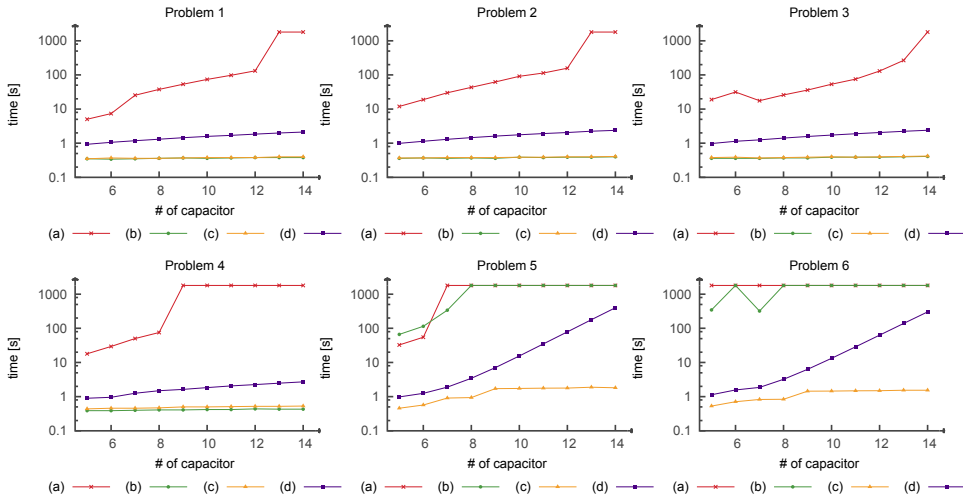


Figure 7.10: CPU times for the 4 configurations, for problems in each of the 6 load profiles.

switched on. This problem can be seen as a temporal planning problem as formulated in (Campion et al., 2013). For the purposes of this thesis we consider a simplified problem, where the objective is to produce more power than the demand for a day. For simplicity we neglect the cost model. By contrast with the example used in Section 4.2, we have multiple units, therefore we need to generalise the actions `(ramp-down)` and `(ramp-up)` in Figure 4.5 to take as an additional parameter the unit that we are considering. We also include the actions `(switch-on ?unit)` and `(switch-off ?unit)` that model the fixed switch on and switch off times. When the unit is switched on, its output is set to its minimum available level.

The initial state is characterised by the demand behaviour through time, modelled with TIFs, to which the planner needs to respond.

Rover Domain

This is a variant of the numeric rover domain in which the rover is equipped with a solar panel that gives energy to the rover according to the exposure to the sun (modelled with TIFs). To perform every action, the rover needs to have at least the same amount of power coming from the solar panel as is consumed. In case of low power, a battery can be used to provide further energy, but with limited capacity. This problem involves real planning alongside the background effects. Actions have initial effects on the available energy, but also release the demand on the power supply at their end points.

Temperature domain

This is a PDDL version of the problem presented in (Ono et al., 2012), where the objective is to maintain the temperature of a smart home within the user's preferences. The temperature is subject to weather conditions, but can be regulated by means of electrochromic windows (*dynamic windows*), that can be tinted to a spectrum of opacities. These problems have both upper and lower bounds, but relatively slow changes over the period of the plan. Effects are at the ends of the actions.

Skier domain

A skier descends along a track at constant velocity, but the track is not always straight, so the problem for the skier is to decide when to turn left or right to arrive at the goal without going off track. This problem also has upper and lower bounds (the two sides of the piste), but the track oscillates more frequently than the temperature bound in the Temperature domain. Effects are at the ends of the actions.

Supercharge domain

This domain is modelled as pathological domain where it is preferable to choose actions that have a longer casual chain rather then actions immediately applicable. In this domain there is a lower bound variable that is incremented in unit steps at unit time intervals for k steps. The controlled variable `(x)`, has to be kept above the lower bound and there are n actions available, A_i for $i = 1...n$, where A_i has a duration of i and increases `(x)` by i. There is also a *supercharge* action that achieves the precondition of a further action that increases `(x)` by

a large amount; the supercharge action takes more than 1 time unit. The durations of the actions A_i mean that the earliest effect available to the planner is the smallest increase. A variant problem places a short period at the start of the plan in which the lower bound is constant, so that the supercharge action can be deployed, and there is a final increase in the lower bound by more than n. This means that the variant problem requires the plan to execute the supercharge and the large increase in the period of grace at the start of the plan, or else to acquire a period of grace later in the plan by increasing `(x)` by enough to allow time to apply supercharge.

Crazy Curves domain

This domain models an abstractly changing pair of bounds, within which a controlled variable must be maintained. However, the curves are generated using random processes that ensure a mix of temporal densities in the TIFs over the plan.

7.2.2 Aims of the Study

To evaluate the performance of POPF-TIF, we run the planner and we measure the CPU time spent to find the solutions and the number of states evaluated.

First, we want to evaluate how each of the new features introduced (the two mixed search strategies without lookahead, and EHC+BFS with one lookahead) compares with the baseline planner, which is POPF-TIF using a heuristic evaluation with 0 lookahead and a search strategy that combines EHC and BFS when the first algorithm fails. We refer to this configuration as "Standard POPF". In Figure 7.11 we report the comparison between the execution times of these three strategies and the baseline planner.

We used the Wilcoxon signed-rank test (Wilcoxon, 1992) to measure the significance of the difference in performance obtained using each of the three strategies as opposed to using the baseline planner. For the 210 problems we calculate the absolute difference in execution time of the baseline planner and the new strategies. We exclude the problems for which there is no difference and the remaining N_r problems are ordered from the smallest to the largest absolute values $|x_{2,i} - x_{1,i}|$. A rank, R_i, is assigned to each pair.

The test statistic W is calculated as

$$W = \left| \sum_{i=1}^{N_r} \left[sgn\left(x_{2,i} - x_{1,i}\right) \cdot R_i \right] \right|. \tag{7.3}$$

Then, the *z-score* is calculated as

$$z = \frac{W - 0.5}{\sigma_W}, \tag{7.4}$$

where

$$\sigma_W = \sqrt{\frac{N_r \left(N_r + 1\right) \left(2N_r + 1\right)}{6}} \tag{7.5}$$

This procedure gives a *z-score* of $z = 6.51$ for MS1, $z = 6.21$ for MS2, and $z = 6.74$ for the lookahead heuristics, meaning that each of the three strategies is independently statistically significant better than the baseline planner on the benchmark considered.

Subsequently we can combine the various search strategies with different lookaheads in

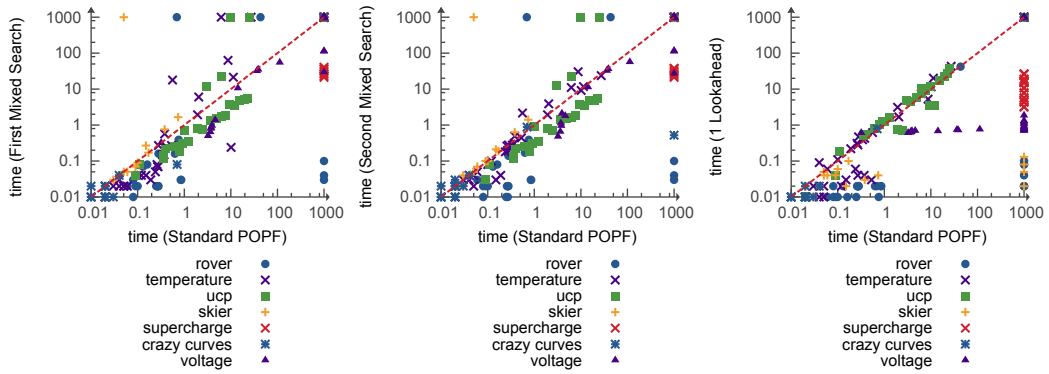


Figure 7.11: Plots showing time performance comparisons for 1 lookahead search and standard POPF. Standard POPF solves 135 problems, MS1 solves 142, MS2 solves 144 and 1 lookahead solves 181.

the heuristic. For each problem we run the planner POPF-TIF in different configurations. The use of the lookahead heuristic is orthogonal to the search strategy used. We combine the lookahead heuristic using 13 different lookaheads of 0,1,2,..11 and 100 with:

- EHC;
- EHC+BFS (in the following we will refer to this search strategy as BFS);
- MS1;
- MS2.

In total we have 52 configurations. The aims of this evaluation are to:

(i) compare all of the different search strategies with the baseline planner;

(ii) determine whether there is a particular combination of lookahead and search algorithm that has significantly better results with respect to the others;

(iii) determine the impact of the different lookaheads on the search strategy;

(iv) determine which lookahead to use according to the characteristics of the domain, in case there is not clearly a best lookahead.

7.2.3 Aggregate Results for all Domains

We want to analyse how the combination of search and heuristics can influence the performance of POPF-TIF. In Figure 7.12 the cumulative numbers of problems solved by the strategies with various lookaheads are plotted against the time taken. In particular, we plot results for lookaheads 0, 1, 2, 3 and best lookahead. This is defined in terms of the region with the biggest area under the corresponding curve. From this figure we can extract various data, which we summarise in Table 7.2. In particular, we report the area under the curves (with the x-axis in logarithmic scale), the maximum number of problems solved, the number of problems solved within 3 seconds, and the CPU time consumed to solve 80% of

Aggregate Results																
Area of behind the cumulative curves																
	EHC				BFS				MS1				MS2			
	0	1	2	b	0	1	2	b	0	1	2	b	0	1	2	b
CC	0.00	0.90	0.96	0.96	0.91	0.92	0.96	0.98	0.92	0.95	0.96	**0.99**	0.93	0.95	0.96	0.98
R	0.00	0.83	0.68	0.83	0.49	0.85	0.69	**0.85**	0.64	0.83	0.68	0.83	0.64	0.83	0.67	0.83
S	0.03	0.58	0.54	0.58	0.66	0.81	0.80	0.83	0.63	0.72	0.82	**0.90**	0.63	0.83	0.82	0.87
SC	0.00	0.00	0.00	0.31	0.00	0.30	0.38	0.51	0.10	0.00	0.30	0.70	0.09	0.15	0.46	**0.74**
T	0.00	0.32	0.28	0.32	0.64	0.67	0.66	0.74	0.65	0.70	0.69	0.78	0.65	0.70	0.71	**0.79**
U	0.00	0.18	0.26	0.27	0.53	0.53	0.59	**0.59**	0.56	0.46	0.58	0.58	0.56	0.55	0.54	0.56
V	0.03	0.56	0.55	0.56	0.18	0.56	0.55	**0.56**	0.24	0.56	0.55	0.56	0.23	0.56	0.55	0.56
ALL	0.01	0.48	0.46	0.48	0.48	0.66	0.65	0.66	0.53	0.60	0.65	0.66	0.53	0.65	0.67	**0.68**
Total number of problems solved																
	EHC				BFS				MS1				MS2			
	0	1	2	b	0	1	2	b	0	1	2	b	0	1	2	b
CC	0	27	29	29	28	28	29	**30**	28	29	29	**30**	29	29	29	**30**
R	0	26	21	26	20	27	22	**27**	21	26	21	26	21	26	21	26
S	1	18	17	18	22	26	26	**30**	21	24	26	**30**	21	26	26	**30**
SC	0	0	0	10	0	20	23	24	8	0	20	**30**	7	11	26	**30**
T	0	10	9	10	27	27	27	**30**	25	26	26	29	27	27	27	**30**
U	0	7	10	10	27	27	29	**29**	25	20	25	25	25	25	24	25
V	1	26	27	27	11	26	27	27	14	26	27	**28**	14	26	27	27
ALL	2	114	113	114	135	181	183	185	142	151	174	178	144	170	180	**188**
Time spent to solve 80% of the problems																
	EHC				BFS				MS1				MS2			
	0	1	2	b	0	1	2	b	0	1	2	b	0	1	2	b
CC	-	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01
R	-	0.03	-	0.03	-	0.03	-	0.03	-	0.03	-	0.03	-	0.04	-	0.04
S	-	-	-	-	-	0.08	0.13	0.08	-	1.36	0.07	**0.07**	-	0.07	0.13	**0.07**
SC	-	-	-	-	-	-	-	16.27	-	-	-	1.76	-	-	11.05	**1.46**
T	-	-	-	-	8.69	7.46	11.43	0.70	21.49	20.00	22.76	**0.17**	11.10	10.46	8.69	0.17
U	-	-	-	-	15.10	17.52	10.75	10.75	12.22	-	5.92	**5.92**	12.23	15.95	32.20	12.23
V	-	1.17	1.90	1.17	-	**1.13**	1.89	**1.13**	-	1.14	1.91	1.14	-	**1.13**	1.89	**1.13**
ALL	-	-	-	-	-	11.88	8.70	7.04	-	-	10.42	4.62	-	21.37	4.61	**3.69**
Number of problems solved in 3 sec																
	EHC				BFS				MS1				MS2			
	0	1	2	b	0	1	2	b	0	1	2	b	0	1	2	b
CC	0	27	29	29	28	28	29	**30**	28	29	29	**30**	29	29	29	**30**
R	0	26	21	26	20	**26**	22	**26**	21	**26**	21	**26**	21	**26**	21	**26**
S	1	18	17	18	22	26	26	**30**	21	24	26	30	21	26	26	**30**
SC	0	0	0	10	0	0	6	18	0	0	0	26	0	0	16	**28**
T	0	10	9	10	22	22	20	**28**	22	22	22	26	22	22	22	26
U	0	6	8	8	16	14	18	18	18	14	**20**	**20**	18	18	18	18
V	1	26	26	**26**	2	**26**	**26**	**26**	8	**26**	**26**	**26**	8	**26**	**26**	**26**
ALL	2	112	110	112	108	142	144	150	116	142	144	152	118	146	158	**160**

Table 7.2: Summary data for all the domains.

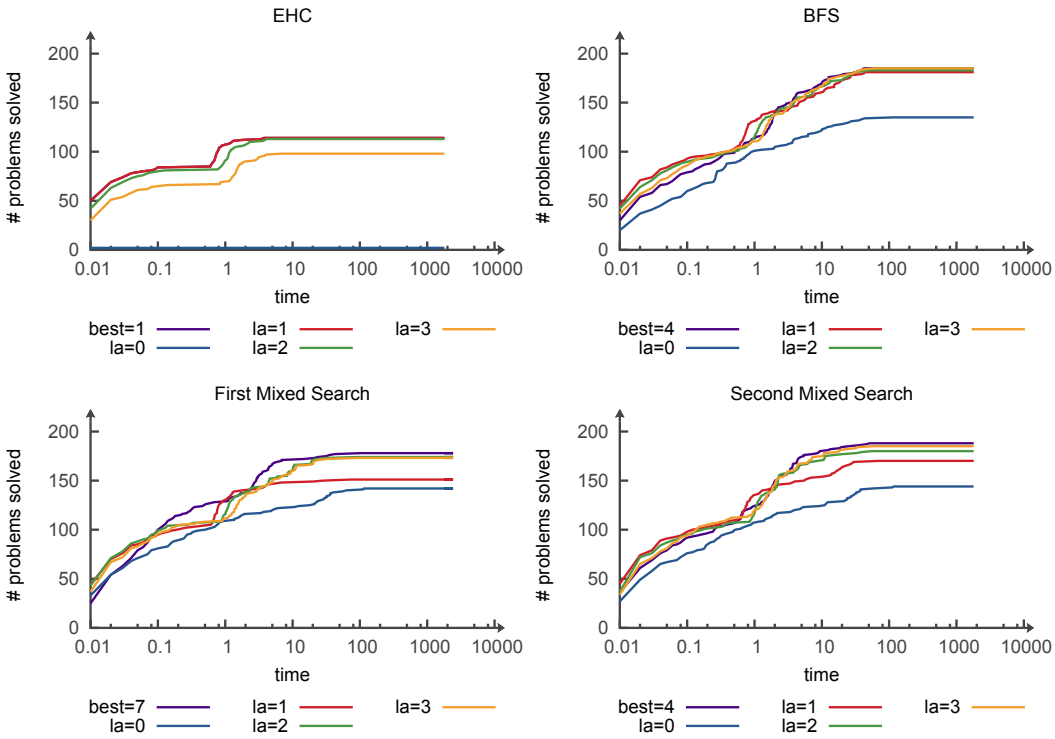


Figure 7.12: Cumulative problems solved for 4 search strategies over time in seconds, using representative lookaheads

	EHC	BFS	MS1	MS2
CC	2	3	3	3
R	1	1	1	1
S	1	8	4	5
SC	100	3	7	10
U	2	2	2	0
T	1	8	8	8
V	2	2	5	2
ALL	1	4	7	4

Table 7.3: Best lookahead for the various domains and various search strategies.

the problems. For readability, we report only the numeric values for 0, 1, 2 and the best lookahead for each domain. Complete data are reported in Appendix E.

The results in the table show that all the four search strategies gain benefit from the lookahead. The biggest improvement is observed when moving from lookahead 0 to lookahead 1 for all the search strategies. Using the heuristic with multiple lookaheads further improves the performance of POPF-TIF, both in terms of coverage and speed. In terms of coverage, this improvement does not necessarily increase with the number of lookaheads, but we observe a significant drop in speed. We also observe that different domains and different search strategies have a different best lookahead, as reported in Table 7.3. The search strategy most affected by the lookahead is EHC, which goes from solving only 2 problems without lookahead, to 114 problems with 1 lookahead, which is what we expect, since the EHC is a local search and it has a limited backtrack mechanism that can recover from dead-ends. The role of the lookahead is to identify and prevent the possible dead-ends

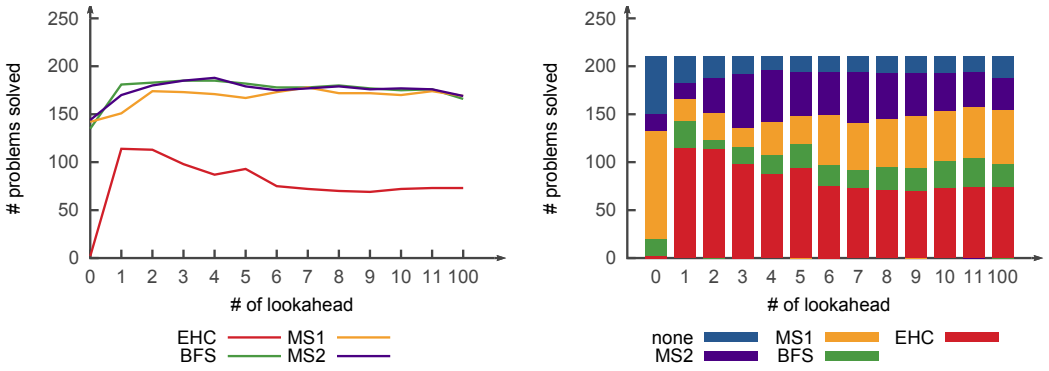


Figure 7.13: Numbers of problems solved with varying lookahead for all the domains, and percentage of problems for which a search strategy is the best, when varying the lookahead.

introduced by TIFs. This phenomenon can be also observed in Figure 7.13, where the left plot represents the number of problems solved as a function of the number of lookaheads used in the heuristic evaluation for the different search strategies. This plot also shows that for a lookahead bigger than 2, the EHC is losing performance, while BFS, MS1 and MS2 tend to maintain their performance with longer lookaheads. In the right plot of Figure 7.13, each bar is divided according to the number of problems for which each search strategy is the best in terms of execution time. Excluding the EHC, either MS1 or MS2 appear to be the best strategies more frequently than BFS. When we consider the coverage and the speed, looking at the results in Table 7.3, the best search strategy is the second mixed strategy MS2, although the improvement with respect to the others is marginal. In terms of coverage, MS2 solves only 3 problems more than BFS and 10 more problems than MS1. The time required to solve 80% of the problems is lower for the two mixed search strategies (MS1 requires 4.62 seconds, while MS2 3.96 seconds) against the 7.04 seconds of BFS.

7.2.4 How to Choose the Lookahead

We now want to study whether it is possible to determine the best lookahead knowing the particular domain that we are solving.

Presence of Double Bounds

Figure 7.14 shows the heuristic evaluation of states for the supercharge domain, which is characterised by only a lower bound, using the first problem variant (in the left plot of the figure) in which the uncontrollable fluent changes incrementally at every time-step. The search is EHC+BFS, with the switch from EHC to BFS marked by the vertical dotted line in the heuristic plot. This shows the values of the states evaluated during the search, with different lookaheads. As it can be seen, the longer lookahead gives a more accurate assessment for most states and also leads to the solution being found much faster (42 states are evaluated for maximum lookahead compared with about 1000 for no lookahead and about 80 states with 1 lookahead). For the second variant problem, shown in Figure 7.15, a 0 lookahead leads to repeated discovery of dead-ends (as shown in the oscillating plot over the 1000-100000 state period), while a 1 lookahead meets significant difficulty in the 100-

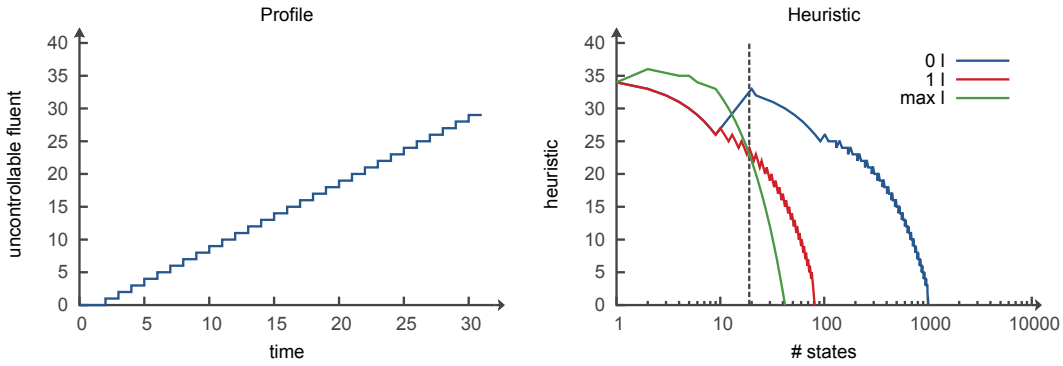


Figure 7.14: Supercharge domain, first variant of the problem.

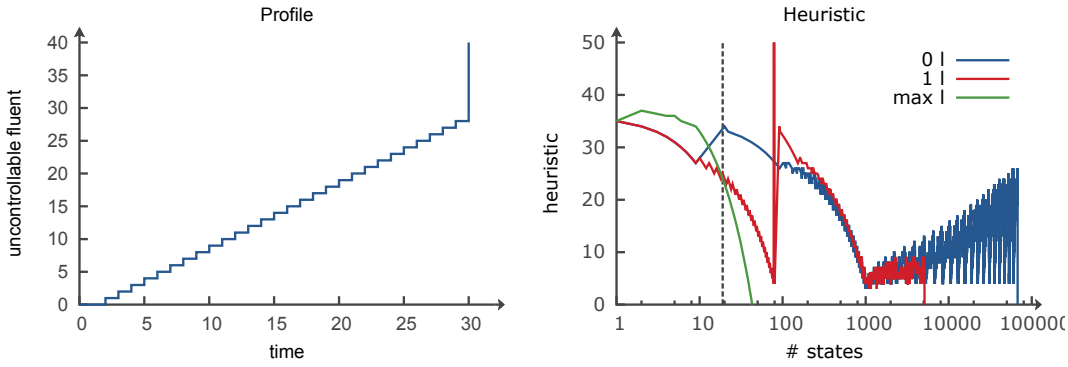


Figure 7.15: Supercharge domain, second variant problem.

10000 state period. The maximum lookahead gives the expected plan in 43 state evaluations. Note that the heuristic counts the TIFs that remain to be executed as well as the actions added to the plan, so the low cost states are close to the end of the timeline.

A broader evaluation over less contrived problems shows that the pattern is not always so simple. In particular, Figure 7.17 shows that for the Skier and Temperature domains, increasing lookahead can lead to deterioration in performance, with a peak at a lookahead of 3. This is because in these problems there is both an upper and a lower bound. A longer lookahead can lead the planner to consider actions that anticipate changes in one of the bounds at a point before a change in the other bound will undermine the corrective action taken. For example, if the upper bound will decrease in 10 minutes, but the lower bound will increase in 5 minutes and fall back to its current level in 7 minutes, then decreasing the controlled variable should be delayed until after 7 minutes, but will appear helpful immediately (see Figure 7.16).

Density of TIFs

Another hypothesis that we want to verify is the dependency of the lookahead on the density of the TIFs with respect to the length of the action that must be taken to resolve a dead-end caused by a TIF. The intuition behind this hypothesis is shown in Figure 7.18. We consider 4 problems of the UCP domain with increasing density of TIFs, and we solve these problems with different lookaheads using the EHC algorithm. As the results reported in Table 7.4

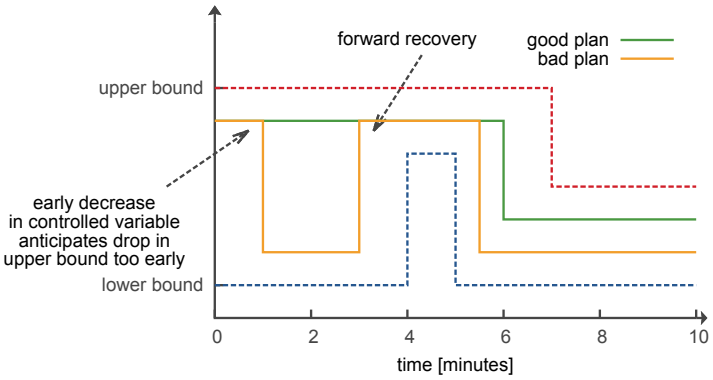


Figure 7.16: Simple example problem in which longer lookahead can be damaging.

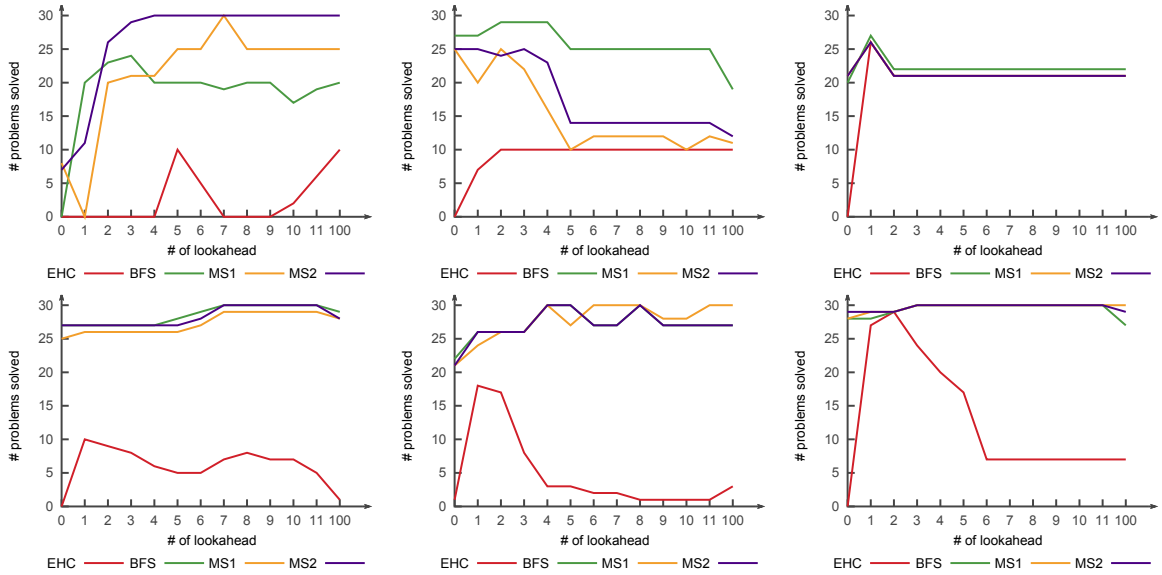


Figure 7.17: Number of problems solved with varying lookahead: Supercharge, UCP and Rovers domains (left to right, top row), Temperature, Skier and Crazy Curves domains (left to right, bottom row)

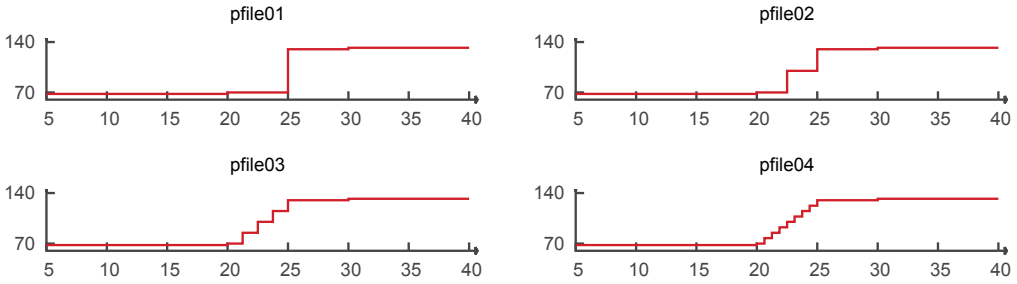


Figure 7.18: Different problem files with different TIF density.

Lookahead	pfile01	pfile02	pfile03	pfile04
0				
1	✓			
2	✓	✓		
3	✓	✓		
4	✓	✓	✓	
5	✓	✓	✓	
6	✓	✓	✓	✓

Table 7.4: Problems solved by POPF-TIF with different lookaheads in the heuristic evaluation, with EHC.

show, the more dense the TIFs, the more lookahead is required in the heuristic evaluation.

In practice, it is not so easy to correlate the density of TIFs with the best lookahead, because several factors are involved, such as the number of actions required to resolve a TIF, the presence of several actions with different duration constraints, the presence of multiple bounds and the presence of non-homogenous TIFs. These make it hard to predict in advance which lookahead works best.

7.2.5 Comparison with Other Planners

We compare the results of POPF-TIF with other planners. Although only UPMurphi is capable of dealing directly with TIFs, it is possible to translate the domains to use only

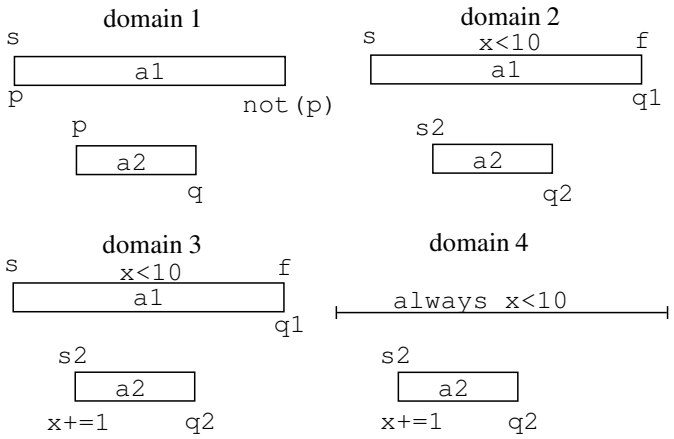


Figure 7.19: Domains used to study planners capabilities.

Planner	domain 1	domain 2	domain 3	domain 4
UPMurphi	✓			
LPG-td		✓	✗	
SGPlan	✓			
TILSapa	✓			
MIPS-XXL	✗	✗	✗	✗

Table 7.5: Domains solved by the planners. ✓: planner solves the domain, ✗: the planner produces an invalid output and blank: no plan produced.

TILs. Each TIF is replaced by two TILs that open and close a time window in which an action that changes the numeric fluent of the TIF is forced to be applied, as discussed in Section 4.1.1.

All these domains share the need for concurrency between actions, in particular with those actions that have `over all` constraints. For this reason, we tried the planners with simplified problems (see Figure 7.19), to test their ability to handle the required concurrency (see Table 7.5).

The first domain is only propositional and requires two actions to be executed concurrently: `a1` is a durative action that adds the precondition of `a2` at the beginning of its execution and deletes it at the end. This domain is solved correctly only by UPMurphi (Della Penna et al., 2009), SGPlan (Chen et al., 2006) and TILSapa (Kavuluri and U, 2004), while LPG-td (Gerevini et al., 2006) results in an empty plan and MIPS-XXL (Edelkamp and Jabbar, 2008) produces an invalid plan with the two actions in sequence. In the second domain we have two actions: the first is the action that simulates a trajectory constraint, imposing a numeric `over all` condition on a fluent, and the second action, `a2`, adds the goal, but does not interact with the `over all` constraint. In this case only LPG-td produces a valid plan, while MIPS-XXL outputs an invalid plan and UPMurphi[1], TILSapa and SGPlan do not produce any plan. The third domain has an interaction between action `a2` and the `over all` constraint, resulting in the failure of LPG-td. In the last domain the action `a1` is replaced with an `always` constraint using PDDL3. The only planners that support PDDL3 are SGPlan and MIPS-XXL, but neither of them produces a valid plan. CPT (Vidal and Geffner, 2006) could deal with temporal concurrency but not with the metric planner. We have not included the planner TFD (Eyerich et al., 2009) because it does not support TILs and the required concurrency.

These results show that, although a number of planners are developed to handle generic domains, their capabilities are not sufficient to handle more complex temporal numeric problems that require limited resources and concurrency.

[1]Since the action `a2` is needed to simulate the behaviour of a TIF and UPMurphi handles TIFs, for this planner we also tested with `a2` replaced with a TIF.

Chapter 8

Conclusions

In this thesis we have introduced the AC voltage control problem as a planning problem. In this domain, electrical loads and generation units consume and produce electrical energy with profiles that vary with time and modify the conditions of the network. Accurate predictions can be made to model their behaviours. The aim is to maintain the voltage of the network within bounds, to guarantee the quality of the electrical energy supplied and the safety of the operational conditions. Actions can change the configurations of network components, causing the voltage to increase or decrease at the busbars. We identified two features that make this a challenging domain for the *state of the art* metric temporal planners: the first one is given by the presence of predictable numeric events, that we model with timed initial fluents, while the other is given by the presence of non-linear numeric effects. We have formulated and addressed these challenges by extending a planner, POPF2, to POPF-TIF.

We have defined a class of problems, called *Bounded Trajectory Management Problems* (BTMPs), characterised by having trajectory constraints influenced by predictable external events. To provide heuristic information in the presence of TIFs, we added to the TRPG heuristic a lookahead mechanism that allows the planner to calculate the effect of the next TIF on the maintenance of the trajectory constraints. This approach is general to the BTMP class. In addition we provide POPF-TIF with two alternative search algorithms that can exploit the TIF-structured nature of the problem.

The non-linear effects of the AC power flow are calculated as semantic attachments that are modelled by introducing an encapsulated type, Network, allowing us to efficiently capture the network-wide effects of local changes to controllable elements of the network. Managing AC power flow is computationally expensive, so we limited the use of accurate calculations of voltages to the state progression phase of search. During the heuristic computation, an abstraction of the network type is used that is based on a first order approximation of the effects of the actions. In some problems, network state changes can undermine the validity of the first order approximations generated in the initial state so we also considered recalculation of the first order approximation in each state as it is evaluated. This more informed approximation of the effects of actions turned out to be of value for some of the problems considered. However, it is a computationally expensive operation and not justified by our experiments.

We also described the application of our approach to the AC voltage control of the

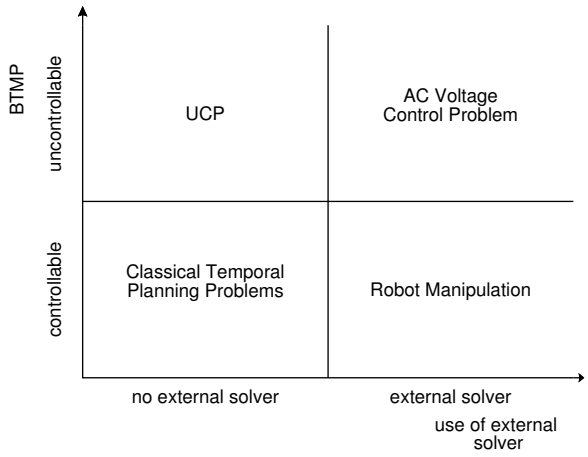


Figure 8.1: Reminder of the two-dimensional space created by BTMPs and problems requiring external solver integration

AuRA-NMS case study network. This network models some of the assumptions of future distribution networks, characterised by the presence of distributed generators. In our experiments, a reactive control approach does not perform well, while the planning approach always finds solutions that satisfy the voltage limits of each busbar of the network. We studied the limitations of the planner, considering networks increasing in size as the number of passive elements (such as busbars and lines) increased, and then as the number of active control points (transformers, capacitors, etc) increased. The number of passive elements is not an issue for the planner: we observed an increase in the running time, due to the time spent by the external solver to compute the solution of the power flow equations, but this does not affect the number of states evaluated by the planner. In contrast, performance on networks with an increasing number of control points is affected, but it remains acceptable for networks with tens of control elements and many tens of busbars.

The approach, although inspired by the AC voltage control problem, is more general, offering a generic solution to BTMPs and a strategy for tackling the integration of an external solver. It should be remembered that the AC voltage control problem is an example of a combination of the two dimensions of the space in Figure 8.1 and POPF-TIF is the only planner that can deal with problems in the top right quadrant.

The lookahead heuristic and the alternative search strategies were evaluated over a range of BTMPs taken from various domains. From this empirical study we can conclude that the multiple lookahead heuristic is essential to solve more problems. Also the mixed search strategies contribute extending the coverage of the problems solved, but their benefit is to reduce the time spent by POPF-TIF to solve the problems. The combination of mixed search strategy (MS2) and multiple lookahead gives the best performance in terms of both coverage and speed.

8.1 Future Work

The work presented in this thesis offers different directions for future work, which include extending domain independent metric temporal planning, Planning Modulo Theories and

application of planning in power systems. Some of these directions are outlined in the remainder of this section.

8.1.1 Future Work on Lookahead and Numeric Exogenous Events

The performance of the multiple lookahead heuristic depends on how far we look ahead. One interesting open issue is to provide an automated way to choose this parameter. Contrary to our first intuition that a greater lookahead provides the better performance, this parameter seems to be dependent on many different factors, for example the presence of various bounds and the density of TIFs. We can exploit these features to feed them into some learning algorithms, such as Artificial Neural Networks and Decision Trees (Mohri et al., 2012) to automatically select a suitable lookahead. Recently a number of planners were developed to integrate learning algorithms to determine the best configuration of the planner or the best planner among a portfolio. Fast Downward Cedalion (Seipp et al., 2014a), Fast Downward SMAC (Seipp et al., 2014b), LLAMA (Virseda et al., 2014), and MIPPlan (Nùñez et al., 2014) are just few examples of these planners.

Still in the context of BTMPs, another interesting topic for future investigation is given by the approximation of the background events that can be expressed as continuous changes through the time. The timed initial fluents that we have included allow us to model independent numeric changes as discretised step functions. A refinement of this approximation could be to consider linear-piecewise approximation between two time-points. These kind of effects are similar to the processes introduced in PDDL+, which are transition function with continuous changes that are automatically triggered by some preconditions that become satisfied.

8.1.2 Future Work on Abstraction of Semantic Attachments

In this thesis we have also considered the coupling of a declarative planner with external solvers for the determination of non-linear numeric effects. We extended the PDDL language with an *encapsulated type* which hides its particular structure from the planner. The full semantics and syntax of this language is out of the scope of this thesis and it is not provided, but it would be of interest to the planning community. Moreover the abstraction provided in this thesis to model the effect of the network depends on the external solver used and it is left to the model designer. An interesting extension of this work is towards the identification of the principles for the abstraction of the *encapsulated types*. This steps towards the goals of Planning Modulo Theories (Gregory et al., 2012).

8.1.3 Future Work on Applications

From the application point of view, we can extend our model of the AC voltage control problem to directly handle the uncertainties of the load and generation profiles. Different approaches can be used. In the paper (Ono et al., 2012) the authors use a risk-sensitive model-based plan executive in order to control the room temperature in an energy efficient way. They use *chance constraints* that specify a lower bound on the probability of failing, then they reformulate the stochastic constraints in deterministic terms on nominal states.

The problem can be translated in terms of a fixed-risk planning problem that feeds an iterative algorithm *(IRA-CCQSP)*. Starting with a uniform risk allocation, the algorithm solves the problem with fixed-risk at each step and reallocates the risk in the next step. We could use this approach to handle the uncertainties in the demand, using TIFs to capture risks bounds and use our model for the fixed-risk planning phase. Another approach is the plan-based policy learning algorithm described in (Fox et al., 2012). In this paper, planning for deterministic problems and Monte Carlo sampling for policy learning are combined to build effective policies for battery switching on the presence of stochastic load profiles. We could use our approach to enhance the planning for plan-based policy learning.

Future electricity networks also offer other problems that will need to be tackled. Amongst these is the *power flow management* problem. This is the problem of deciding how to constrain the electrical power flow in the network, by curtailing the output of generators or changing the topology of the network, to satisfy trajectory constraints such as thermal constraints on the power flowing through the lines. In this problem a richer model of temporal constraints might be exploited, to allow small violations that do not exceed a given period of time, using the ideas discussed in Section 4.5.

Appendix A

A More Detailed Explanation of the Power Flow Equations

The computation of the voltage at each node of the network, for specified loads and generation configurations, is called the *power flow* (or *load flow*) problem. The nodes of the networks are called *busbars* (or *buses*). They are strips of electrical conductors with the main purpose of conducting electricity between two branches (*lines* and *cables*) of the network.

There are four variables associated with a busbar k: voltage magnitude, $\mathrm{V_k}$, phase angle, δ_k, real power, P_k, and reactive power, Q_k. Real (sometimes called *active*) power and reactive power are two complementary forms of power. Real power is the quantity associated with the total energy absorbed by a load during a time interval of one cycle of the sinusoidal voltage.

The goal of a power-flow study is to obtain complete voltage angle and magnitude information for each bus in a power system for specified load and generator real power and voltage conditions. Once this information is known, real and reactive power flow on each branch as well as generator reactive power output can be analytically determined. Due to the non-linear nature of this problem, numerical methods are employed to obtain a solution that is within an acceptable tolerance.

In general, the total power delivered to a busbar k is divided into two parts, the generation and the load:

$$\begin{aligned} P_k &= P_{Gk} - P_{Lk} \\ Q_k &= Q_{Gk} - Q_{Lk} \end{aligned} \tag{A.1}$$

For the power flow computation two of these variables are given as input data, while the other two are computed. Each busbar k is then categorised into different bus types, according to the input variables:

- **Slack Busbar (or swing busbar):** one bus in the network is defined to balance the active and reactive power of the system. The voltage magnitude[1] $\mathrm{V_k}$ and the phase

[1] Note that the voltage as complex quantity is indicated in italics V, while its magnitude is in regular font V.

angle δ_k are assumed to be known and they are generally $\mathrm{V_k} = 1$ pu and $\delta_k = 0$. The power flow equations are over-constrained (the matrix of coefficients is singular), so the slack bus is needed to provide an additional degree of freedom. It corresponds physically to the freedom of choice about initial phase angle.

- **Load Busbar (or PQ):** P_k and Q_k are given as input, while $\mathrm{V_k}$ and δ_k are calculated by the power flow.
- **Voltage controlled Busbar (or PV):** for this busbar P_k and $\mathrm{V_k}$ are input data and Q_k and δ_k are computed.

In order to perform the power flow calculation, the admittance matrix Y_{bus} must be constructed from the line and transformer input data, where Y_{kk} is the sum of admittances of lines connected to bus k, and Y_{kn} is the opposite of the sum of admittances connected between buses k and n (with $k \neq n$).

The steady state AC power flow for the busbar k is then given by:

$$P_k + jQ_k = V_k \left[\sum_{n=1}^{N} Y_{kn} V_n \right]^* \tag{A.2}$$

where V_k is the complex voltage $\mathrm{V_k} e^{j\delta_k}$ and j is the complex square root of -1. In rectangular coordinates the power flow equations are expressed as:

$$\begin{aligned} P_k &= \mathrm{V_k} \sum_{n=1}^{N} \mathrm{V_n} \left[G_{kn} \cos(\delta_k - \delta_n) + B_{kn} \sin(\delta_k - \delta_n) \right] \\ Q_k &= \mathrm{V_k} \sum_{n=1}^{N} \mathrm{V_n} \left[G_{kn} \sin(\delta_k - \delta_n) - B_{kn} \cos(\delta_k - \delta_n) \right] \end{aligned} \tag{A.3}$$

where $Y_{kn} = \mathrm{Y_{kn}} \mathrm{e}^{\mathrm{j}\theta_{\mathrm{kn}}} = G_{kn} + jB_{kn}$. The quantities G_{kn} and B_{kn} are called respectively reactance and susceptance of the line between busbar k and n.

The generalisation of the power flow equations (A.3), which includes transformers, is modelled by modifying the admittance matrix Y. Given a transformer connected between busbar k and n:

$$\begin{aligned} Y'_{kk} &= Y_{kk} - Y_{kn} + Y_{kn}/|t_{kn}|^2, \\ Y'_{kn} &= Y_{kn}/t^*_{kn}, \\ Y'_{nk} &= Y_{nk}/t_{kn} \end{aligned} \tag{A.4}$$

Due to the non-linearity of the power flow equations, their integration through traditional decision-support tools is very limited. Instead, different linear approximations can be applied in situations in which it is possible to consider only the active power. This is the case of transmission networks, where the ratio G_{kn}/B_{kn} is very small, the phase angle difference is small enough and the voltage magnitudes are close to 1.0 pu, so that the Equations A.3 reduce to:

$$P_k = \sum_{n=1}^{N} -B_{kn}(\delta_k - \delta_n) \tag{A.5}$$

Appendix B

The PDDL2.2 Domain Description for the AC Voltage Control Problem

In the following, we present the AC voltage control domain, as used by POPF-TIF. We introduce some new syntax to allow us to define the external (encapsulated) type, Network, and a collection of external functions which form the interface between the planner and external solver. In fact, this syntax can be pre-processed into PDDL2.2 with TIFs, as is shown in the domain model presented in the Appendix D. Our approach is to do this pre-processing, but we show the intended domain representation as a first step towards a clean interface between POPF-TIF and its external solvers.

```
(define (domain APS)
(:requirements :strips :typing :fluents :universal-preconditions
:disjunctive-preconditions :conditional-effects :duration-inequalities
:durative-actions :timed-initial-literals
:preferences :constraints)
(:external-type Network)
(:types
    gen - object
    bus - object
    device - object
    tap - device
    capacitor - device)
(:predicates (is-available ?t - tap) (is-before-end) (is-end) (S) (F)
 (is-active ?c - capacitor) (is-not-active ?c - capacitor)
 (is-not-increasing ?t - tap) (is-not-decreasing ?t - tap)
 (is-increasing ?t - tap) (is-decreasing ?t - tap)
 (can-increase ?t - tap) (can-decrease ?t - tap))
 (:functions
    (p-level ?l - load)
    (q-level ?l - load)
```

```
    (tap-level ?t - tap)
    (gen-p-level ?g - gen)
    (gen-q-level ?g - gen)
    (maximum-voltage ?b - bus)
    (minimum-voltage ?b - bus)
    (slack-period ?t - device)
    (max-tap-level ?t - tap)
    (min-tap-level ?t - tap))
(:external-functions
    (step-effect ?t - tap)
    (capacitor-effect ?c - capacitor)
    (voltage ?b - busbar))
(:constants thenetwork - Network)

(:durative-action step-down-tap
:parameters (?t - tap)
:duration (= ?duration 0.1)
:condition (and
    (at start (is-not-increasing ?t))
    (at start (is-available ?t))
    (at start  (> (tap-level ?t) (min-tap-level ?t))))
:effect (and
    (at start (not (is-not-decreasing ?t)))
    (at start (not (is-available ?t)))
    (at end (is-available ?t))
    (at end (is-decreasing ?t))
    (at end (decrease (tap-level ?t) 1))
    (at end (decrease (thenetwork) (step-down-effect ?t)))))

(:durative-action release-decreasing
:parameters (?t - tap)
:duration (= ?duration (slack-period ?t))
:condition (and (at start (is-decreasing ?t)))
:effect (and (at end (is-not-decreasing ?t))
    (at start (not (is-decreasing ?t)))))

(:durative-action release-increasing
:parameters (?t - tap)
:duration (= ?duration (slack-period ?t))
:condition (and (at start (is-increasing ?t)))
:effect (and (at end (is-not-increasing ?t))
    (at start (not (is-increasing ?t)))))

(:durative-action step-up-tap
:parameters (?t - tap)
:duration (= ?duration 0.1)
:condition (and
    (at start (is-not-decreasing ?t))
```

```
    (at start  (< (tap-level ?t) (max-tap-level ?t)))
    (at start (is-available ?t)))
:effect (and
    (at start (not (is-not-increasing ?t)))
    (at end (is-increasing ?t))
    (at start (not (is-available ?t)))
    (at end (is-available ?t))
    (at end (increase (tap-level ?t) 1))
    (at end (increase (thenetwork) (step-up-effect ?t)))))

(:durative-action release-capacitor
:parameters (?t - capacitor)
:duration (= ?duration 3)
:condition (and (at start (is-available ?t)))
:effect (and (at end (is-available ?t))
    (at start (not (is-available ?t)))))

(:durative-action activate-capacitor
:parameters (?c - capacitor)
:duration (= ?duration 0.1)
:condition (and
    (at start (is-not-active ?c))
    (at start (is-available ?c)))
:effect (and
    (at start (not (is-not-active ?c)))
    (at start (not (is-available ?c)))
    (at end (is-active ?c))
    (at end (increase (thenetwork) (capacitor-effect ?t)))))

(:durative-action deactivate-capacitor
:parameters (?c - capacitor)
:duration (= ?duration 0.1)
:condition (and
    (at start (is-active ?c))
    (at start (is-available ?c)))
:effect (and
    (at start (not (is-active ?c)))
    (at start (not (is-available ?c)))
    (at end (is-not-active ?c))
    (at end (decrease (thenetwork) (capacitor-effect ?t)))))

(:durative-action constraint-check
:parameters ()
:duration (<= ?duration 1000)
:condition (and
    (at start (S))
    (at end (F))
    (over all  (forall (?b - Bus) (and
```

```
        (<= (thenetwork.voltage ?b) (maximum-voltage ?b))
        (>= (thenetwork.voltage ?b) (minimum-voltage ?b))))))
:effect (and (at end (is-end)))))
```

Appendix C

A Small Problem Instance for the AC Voltage Control Problem

```
(define (problem APS-problem)
(:domain APS)
(:objects
    busbar1  - bus
    load0 - load
    tap0  - tap
)

(:init
    (S)
    (at 0.1 (not (S)))
    (at 24 (F))
    (= (tap-level tap0) 0)
    (is-available tap0)
    (is-not-increasing tap0)
    (is-not-decreasing tap0)
    (= (ipsa-voltage busbar1) 0.95)
    (= (maximum-voltage busbar1) 1)
    (= (minimum-voltage busbar1) 0.94)
    (= (max-tap-level tap0) 4)
    (= (min-tap-level tap0) -9)
    (= (slack-period tap0) 3)
    (= (step-min tap0 busbar1) 0.0031) ; 0.0035)
    (= (step-max tap0 busbar1) -0.0031)
    (= (p-level load0 ) 3.5)
    (= (q-level load0 ) 0.5)
    (at 5 (= (p-level load0) 8.5))
    (at 5 (= (q-level load0) 0.5))
```

```
    (at 10 (= (p-level load0) 3.5))
    (at 10 (= (q-level load0) 0.5))
    (at 15 (= (p-level load0) 3.3))
    (at 15 (= (q-level load0) 0.5))
    (at 16 (= (p-level load0) 2))
    (at 16 (= (q-level load0) 0.5))
    (at 20 (= (p-level load0) 5))
    (at 22 (= (p-level load0) 7))
    (at 22 (= (q-level load0) 0.5)))

(:goal (and (is-end))))
```

Appendix D

The Translated PDDL2.2 Domain Description for the AC Voltage Control Problem

In the following, we present the PDDL2.2 model of the AC voltage control problem processed from the domain in Appendix B.

```
(define (domain APS)
(:requirements :strips :typing :fluents :universal-preconditions
:disjunctive-preconditions :conditional-effects :duration-inequalities
:durative-actions :timed-initial-literals
:preferences :constraints)
(:external-type Network)
(:types
    bus - object
    device - object
    tap - device
    capacitor - device)
(:predicates (is-available ?t - tap) (is-before-end) (is-end) (S) (F)
 (is-active ?c - capacitor) (is-not-active ?c - capacitor)
 (is-not-increasing ?t - tap) (is-not-decreasing ?t - tap)
 (is-increasing ?t - tap) (is-decreasing ?t - tap)
 (can-increase ?t - tap) (can-decrease ?t - tap))
 (:functions
    (p-level ?l - load)
    (q-level ?l - load)
    (tap-level ?t - tap)
    (gen-p-level ?g - gen)
    (gen-q-level ?g - gen)
    (step-min ?t - device ?b - bus)
    (step-max ?t - device ?b - bus)
    (capacitor-level ?c - capacitor)
    (maximum-voltage ?b - bus)
```

```
    (minimum-voltage ?b - bus)
    (slack-period ?t - device)
    (max-tap-level ?t - tap)
    (min-tap-level ?t - tap)
    (voltage ?b - bus)
    )


(:durative-action step-down-tap
:parameters (?t - tap)
:duration (= ?duration 0.1)
:condition (and
    (at start (is-not-increasing ?t))
    (at start (is-available ?t))
    (at start  (> (tap-level ?t) (min-tap-level ?t))))
:effect (and
    (at start (not (is-not-decreasing ?t)))
    (at start (not (is-available ?t)))
    (at end (is-available ?t))
    (at end (is-decreasing ?t))
    (at end (decrease (tap-level ?t) 1))
    (forall (?b - bus) (and (at end (increase (voltage ?b) (step-min ?t ?b)))))))

(:durative-action release-decreasing
:parameters (?t - tap)
:duration (= ?duration (slack-period ?t))
:condition (and (at start (is-decreasing ?t)))
:effect (and (at end (is-not-decreasing ?t))
    (at start (not (is-decreasing ?t)))))

(:durative-action release-increasing
:parameters (?t - tap)
:duration (= ?duration (slack-period ?t))
:condition (and (at start (is-increasing ?t)))
:effect (and (at end (is-not-increasing ?t))
    (at start (not (is-increasing ?t)))))

(:durative-action step-up-tap
:parameters (?t - tap)
:duration (= ?duration 0.1)
:condition (and
    (at start (is-not-decreasing ?t))
    (at start  (< (tap-level ?t) (max-tap-level ?t)))
    (at start (is-available ?t)))
:effect (and
    (at start (not (is-not-increasing ?t)))
    (at end (is-increasing ?t))
    (at start (not (is-available ?t)))
```

```
    (at end (is-available ?t))
    (at end (increase (tap-level ?t) 1))
    (forall (?b - bus) (and (at end (increase (voltage ?b) (step-max ?t ?b)))))))

(:durative-action release-capacitor
:parameters (?t - capacitor)
:duration (= ?duration 3)
:condition (and (at start (is-available ?t)))
:effect (and (at end (is-available ?t))
    (at start (not (is-available ?t)))))

(:durative-action activate-capacitor
:parameters (?c - capacitor)
:duration (= ?duration 0.1)
:condition (and
    (at start (is-not-active ?c))
    (at start (is-available ?c)))
:effect (and
    (at start (not (is-not-active ?c)))
    (at start (not (is-available ?c)))
    (at end (is-active ?c))
    (forall (?b - bus) (and (at end (increase (voltage ?b) (step-min ?t ?b)))))))

(:durative-action deactivate-capacitor
:parameters (?c - capacitor)
:duration (= ?duration 0.1)
:condition (and
    (at start (is-active ?c))
    (at start (is-available ?c)))
:effect (and
    (at start (not (is-active ?c)))
    (at start (not (is-available ?c)))
    (at end (is-not-active ?c))
    (forall (?b - bus) (and (at end (decrease (voltage ?b) (step-min ?t ?b)))))))

(:durative-action constraint-check
:parameters ()
:duration (<= ?duration 1000)
:condition (and
    (at start (S))
    (at end (F))
    (over all  (forall (?b - Bus) (and
        (<= (voltage ?b) (maximum-voltage ?b))
        (>= (voltage ?b) (minimum-voltage ?b))))))
:effect (and (at end (is-end)))))
```

Appendix E

Results of the Lookahead Analysis

In this appendix we report the complete data of the evaluation study in Section 7.2.

EHC													
Area of behind the cumulative curves													
	0	1	2	3	4	5	6	7	8	9	10	11	100
CC	0.00	0.90	0.96	0.79	0.66	0.56	0.23	0.23	0.23	0.23	0.23	0.23	0.23
R	0.00	0.83	0.68	0.67	0.68	0.67	0.67	0.68	0.68	0.67	0.67	0.67	0.68
S	0.03	0.58	0.54	0.25	0.10	0.10	0.07	0.07	0.03	0.03	0.03	0.03	0.09
SC	0.00	0.00	0.00	0.00	0.00	0.29	0.15	0.00	0.00	0.00	0.07	0.19	0.31
T	0.00	0.32	0.28	0.24	0.19	0.15	0.15	0.21	0.24	0.20	0.20	0.15	0.03
U	0.00	0.18	0.26	0.27	0.26	0.27	0.26	0.26	0.26	0.26	0.26	0.26	0.23
V	0.03	0.56	0.55	0.52	0.51	0.49	0.45	0.44	0.40	0.40	0.40	0.38	0.27
ALL	0.01	0.48	0.46	0.39	0.34	0.36	0.28	0.27	0.26	0.26	0.27	0.27	0.26
Total number of problems solved													
	0	1	2	3	4	5	6	7	8	9	10	11	100
CC	0	27	29	24	20	17	7	7	7	7	7	7	7
R	0	26	21	21	21	21	21	21	21	21	21	21	21
S	1	18	17	8	3	3	2	2	1	1	1	1	3
SC	0	0	0	0	0	10	5	0	0	0	2	6	10
T	0	10	9	8	6	5	5	7	8	7	7	5	1
U	0	7	10	10	10	10	10	10	10	10	10	10	10
V	1	26	27	27	27	27	25	25	23	23	24	23	21
ALL	2	114	113	98	87	93	75	72	70	69	72	73	73
Time spent to solve 80% of the problems													
	0	1	2	3	4	5	6	7	8	9	10	11	100
CC	-	0.01	0.01	0.02	-	-	-	-	-	-	-	-	-
R	-	0.08	-	-	-	-	-	-	-	-	-	-	-
S	-	-	-	-	-	-	-	-	-	-	-	-	-
SC	-	-	-	-	-	-	-	-	-	-	-	-	-
T	-	-	-	-	-	-	-	-	-	-	-	-	-
U	-	-	-	-	-	-	-	-	-	-	-	-	-
V	-	1.25	2.12	3.05	3.76	4.86	8.52	20.94	-	-	105.30	-	-
ALL	-	-	-	-	-	-	-	-	-	-	-	-	-
Number of problems solved in 3 sec													
	0	1	2	3	4	5	6	7	8	9	10	11	100
CC	0	27	29	24	20	17	7	7	7	7	7	7	7
R	0	26	21	21	21	21	21	21	21	21	21	21	21
S	1	18	17	8	3	3	2	2	1	1	1	1	3
SC	0	0	0	0	0	10	5	0	0	0	2	6	10
T	0	10	9	8	6	5	5	7	8	7	7	5	1
U	0	6	8	8	8	8	8	8	8	8	8	8	6
V	1	26	26	24	22	20	18	14	8	6	2	0	0
ALL	2	112	110	92	80	84	66	58	54	50	48	48	48

Table E.1: Summary data for EHC.

BFS													
Area of behind the cumulative curves													
	0	1	2	3	4	5	6	7	8	9	10	11	100
CC	0.91	0.92	0.96	0.98	0.97	0.96	0.94	0.94	0.93	0.92	0.91	0.91	0.83
R	0.49	0.85	0.69	0.69	0.69	0.70	0.69	0.69	0.69	0.69	0.69	0.69	0.70
S	0.66	0.81	0.80	0.77	0.81	0.83	0.75	0.75	0.82	0.75	0.75	0.76	0.74
SC	0.00	0.30	0.38	0.40	0.37	0.47	0.39	0.37	0.38	0.39	0.36	0.45	0.51
T	0.64	0.67	0.66	0.66	0.65	0.66	0.67	0.73	0.74	0.72	0.72	0.71	0.61
U	0.53	0.53	0.59	0.59	0.57	0.50	0.49	0.49	0.49	0.48	0.48	0.48	0.34
V	0.18	0.56	0.55	0.52	0.51	0.49	0.45	0.44	0.40	0.40	0.40	0.38	0.27
ALL	0.48	0.66	0.65	0.65	0.65	0.65	0.62	0.62	0.63	0.62	0.61	0.62	0.56
Total number of problems solved													
	0	1	2	3	4	5	6	7	8	9	10	11	100
CC	1	18	17	8	3	3	2	2	1	1	1	1	3
R	20	27	22	22	22	22	22	22	22	22	22	22	22
S	22	26	26	26	30	30	27	27	30	27	27	27	27
SC	0	20	23	24	20	20	20	19	20	20	17	19	20
T	27	27	27	27	27	28	29	30	30	30	30	30	29
U	27	27	29	29	29	25	25	25	25	25	25	25	19
V	11	26	27	27	27	27	25	25	23	23	24	23	21
ALL	135	181	183	185	185	182	178	178	180	177	175	176	166
Time spent to solve 80% of the problems													
	0	1	2	3	4	5	6	7	8	9	10	11	100
CC	0.01	0.01	0.01	0.01	0.03	0.03	0.05	0.05	0.05	0.06	0.06	0.07	0.15
R	-	0.06	-	-	-	-	-	-	-	-	-	-	-
S	-	0.08	0.13	0.15	0.33	0.30	0.35	0.34	0.27	0.41	0.42	0.44	1.24
SC	-	-	-	16.27	-	-	-	-	-	-	-	-	-
T	8.69	7.46	11.43	12.22	9.82	4.41	1.85	1.09	0.70	0.95	1.24	1.34	6.26
U	15.10	17.52	10.75	18.98	11.14	38.53	43.75	49.91	55.49	62.36	69.36	77.11	-
V	-	1.27	2.24	3.05	3.76	4.94	8.53	20.70	-	-	109.72	-	-
ALL	-	15.31	11.43	10.45	8.96	10.94	22.85	23.89	13.92	20.35	42.80	32.47	-
Number of problems solved in 3 sec													
	0	1	2	3	4	5	6	7	8	9	10	11	100
CC	28	28	29	30	30	30	30	30	30	30	30	30	28
R	20	26	22	22	22	22	22	22	22	22	22	22	22
S	22	26	26	26	30	30	27	27	30	27	27	27	26
SC	0	0	6	6	10	14	8	14	12	12	12	15	18
T	22	22	20	22	22	24	24	26	28	28	28	28	22
U	16	14	18	16	16	14	14	14	14	14	12	12	8
V	2	26	26	24	22	20	18	14	8	4	2	0	0
ALL	108	142	144	144	150	150	140	144	142	134	130	132	122

Table E.2: Summary data for BFS.

MS1													
Area of behind the cumulative curves													
	0	1	2	3	4	5	6	7	8	9	10	11	100
CC	0.92	0.95	0.96	0.99	0.98	0.98	0.96	0.96	0.95	0.94	0.93	0.92	0.89
R	0.64	0.83	0.68	0.67	0.68	0.67	0.68	0.67	0.68	0.67	0.68	0.68	0.68
S	0.63	0.72	0.82	0.80	0.90	0.80	0.88	0.86	0.86	0.80	0.80	0.84	0.81
SC	0.10	0.00	0.30	0.32	0.34	0.60	0.52	0.70	0.64	0.68	0.70	0.70	0.66
T	0.65	0.70	0.69	0.69	0.69	0.69	0.70	0.77	0.78	0.76	0.75	0.74	0.66
U	0.56	0.46	0.58	0.51	0.38	0.26	0.30	0.29	0.29	0.29	0.26	0.29	0.24
V	0.24	0.56	0.55	0.52	0.51	0.51	0.49	0.45	0.43	0.43	0.43	0.42	0.29
ALL	0.53	0.60	0.65	0.64	0.63	0.64	0.64	0.66	0.66	0.65	0.65	0.65	0.60
Total number of problems solved													
	0	1	2	3	4	5	6	7	8	9	10	11	100
CC	28	29	29	30	30	30	30	30	30	30	30	30	30
R	21	26	21	21	21	21	21	21	21	21	21	21	21
S	21	24	26	26	30	27	30	30	30	28	28	30	30
SC	8	0	20	21	21	25	25	30	25	25	25	25	25
T	25	26	26	26	26	26	27	29	29	29	29	29	28
U	25	20	25	22	16	10	12	12	12	12	10	12	11
V	14	26	27	27	27	28	28	26	25	27	27	27	24
ALL	142	151	174	173	171	167	173	178	172	172	170	174	169
Time spent to solve 80% of the problems													
	0	1	2	3	4	5	6	7	8	9	10	11	100
CC	0.01	0.01	0.01	0.01	0.01	0.01	0.03	0.04	0.04	0.04	0.04	0.04	0.09
R	-	0.08	-	-	-	-	-	-	-	-	-	-	-
S	-	1.36	0.07	0.12	0.11	0.24	0.16	0.22	0.22	0.36	0.42	0.30	0.52
SC	-	-	-	-	-	6.98	19.98	2.47	4.59	2.02	1.76	1.82	7.53
T	21.49	20.00	22.76	23.94	25.63	9.85	3.42	0.17	0.18	0.44	0.49	0.57	1.26
U	12.22	-	5.92	-	-	-	-	-	-	-	-	-	-
V	-	1.24	2.27	3.05	3.71	4.80	8.51	21.22	11.68	36.40	46.00	55.45	1212.31
ALL	-	-	18.87	20.30	24.05	-	19.19	5.30	11.68	36.40	107.02	29.54	494.14
Number of problems solved in 3 sec													
	0	1	2	3	4	5	6	7	8	9	10	11	100
CC	28	29	29	30	30	30	30	30	30	30	30	30	30
R	21	26	21	21	21	21	21	21	21	21	21	21	21
S	21	24	26	26	30	27	30	30	30	28	28	30	30
SC	0	0	0	2	6	24	16	26	22	24	24	24	22
T	22	22	22	22	22	24	24	26	26	26	26	26	24
U	18	14	20	18	12	8	8	8	8	8	8	8	6
V	8	26	26	24	22	20	18	14	8	5	2	0	0
ALL	116	142	144	140	142	152	146	152	146	142	140	140	132

Table E.3: Summary data for MS1.

MS2													
Area of behind the cumulative curves													
	0	1	2	3	4	5	6	7	8	9	10	11	100
CC	0.93	0.95	0.96	0.98	0.98	0.98	0.95	0.94	0.93	0.93	0.91	0.92	0.84
R	0.64	0.83	0.67	0.67	0.68	0.68	0.68	0.68	0.68	0.68	0.67	0.68	0.68
S	0.63	0.83	0.82	0.79	0.85	0.87	0.79	0.79	0.85	0.77	0.78	0.77	0.74
SC	0.09	0.15	0.46	0.54	0.61	0.73	0.66	0.68	0.68	0.69	0.71	0.72	0.74
T	0.65	0.70	0.71	0.71	0.70	0.69	0.71	0.79	0.79	0.77	0.76	0.74	0.62
U	0.56	0.55	0.54	0.55	0.48	0.33	0.33	0.33	0.33	0.32	0.32	0.32	0.26
V	0.23	0.56	0.55	0.52	0.51	0.49	0.45	0.44	0.42	0.41	0.41	0.39	0.28
ALL	0.53	0.65	0.67	0.68	0.68	0.67	0.65	0.66	0.66	0.65	0.65	0.64	0.59
Total number of problems solved													
	0	1	2	3	4	5	6	7	8	9	10	11	100
CC	29	29	29	30	30	30	30	30	30	30	30	30	29
R	21	26	21	21	21	21	21	21	21	21	21	21	21
S	21	26	26	26	30	30	27	27	30	27	27	27	27
SC	7	11	26	29	30	30	30	30	30	30	30	30	30
T	27	27	27	27	27	27	28	30	30	30	30	30	28
U	25	25	24	25	23	14	14	14	14	14	14	14	12
V	14	26	27	27	27	27	25	25	24	24	25	24	22
ALL	144	170	180	185	188	179	175	177	179	176	177	176	169
Time spent to solve 80% of the problems													
	0	1	2	3	4	5	6	7	8	9	10	11	100
CC	0.01	0.01	0.01	0.01	0.01	0.01	0.04	0.04	0.06	0.06	0.07	0.06	0.14
R	-	0.08	-	-	-	-	-	-	-	-	-	-	-
S	-	0.07	0.13	0.13	0.28	0.17	0.24	0.28	0.28	0.41	0.41	0.40	1.14
SC	-	-	11.05	10.52	3.66	1.46	1.96	1.68	2.47	1.72	1.74	1.90	2.79
T	11.10	10.46	8.69	6.23	12.42	8.68	3.44	0.27	0.17	0.55	0.63	1.13	7.30
U	12.23	15.95	32.20	18.61	-	-	-	-	-	-	-	-	-
V	-	1.24	2.25	3.04	3.71	4.79	8.52	21.00	20.01	35.89	45.84	105.07	-
ALL	-	28.80	6.18	6.10	4.24	5.27	8.52	6.86	5.05	6.78	6.72	10.52	83.08
Number of problems solved in 3 sec													
	0	1	2	3	4	5	6	7	8	9	10	11	100
CC	29	29	29	30	30	30	30	30	30	30	30	30	28
R	21	26	21	21	21	21	21	21	21	21	21	21	21
S	21	26	26	26	30	30	27	27	30	27	27	27	26
SC	0	0	16	18	22	26	26	26	26	26	28	26	24
T	22	22	22	22	22	22	24	26	26	26	26	26	20
U	18	18	18	18	12	10	10	10	10	10	10	10	6
V	8	26	26	24	22	20	18	14	10	4	2	0	0
ALL	118	146	158	158	156	160	154	154	152	144	143	138	124

Table E.4: Summary data for MS2.

Bibliography

Johannes Aldinger and Johannes Löhr. Planning for Agile Earth Observation Satellites. In *ICAPS Workshop on Planning in Continuous Domains*, pages 9–17, 2013.

Varvara Alimisis and Philip C. Taylor. Zoning Evaluation for Improved Coordinated Automatic Voltage Control. *IEEE Transactions Power Systems*, pages 1–11, 2014.

James F. Allen. Maintaining Knowledge about Temporal Intervals. *Communications of the ACM*, 26(11):832–843, 1983. ISSN 00010782. doi: 10.1145/182.358434.

Christer Bäckström and Bernhard Nebel. Complexity Results for SAS+ Planning. *Computational Intelligence*, 11(4):625–655, 1995. ISSN 0824-7935. doi: 10.1111/j.1467-8640.1995.tb00052.x.

Keith Bell, Amanda Coles, Andrew Coles, Maria Fox, and Derek Long. The Role of AI Planning as a Decision Support Tool in Power Substation Management. *AI Communications*, 22: 37–57, 2009. doi: 10.3233/AIC-2009-0443. URL `http://iospress.metapress.com/index/986kl54667577u5q.pdf`.

Pascal Bercher, Susanne Biundo, Thomas Geier, Thilo Hoernle, Florian Nothdurft, Felix Richter, and Bernard Schattenberg. Plan, Repair, Execute, Explain - How Planning Helps to Assemble your Home Theater. In *International Conference on Automated Planning and Scheduling*, pages 386–394, 2014. URL `http://www.uni-ulm.de/fileadmin/website_uni_ulm/iui.inst.090/Publikationen/2014/Bercher14PlanRepairExecuteExplain.pdf`.

Sara Bernardini, Maria Fox, and Derek Long. Planning the Behaviour of Low-Cost Quadcopters for Surveillance Missions. In *International Conference on Automated Planning and Scheduling*, pages 445–453, 2014. URL `http://www.aaai.org/ocs/index.php/ICAPS/ICAPS14/paper/download/7932/8048`.

Fabio Bignucolo, Roberto Caldon, and Valter Prandoni. Radial MV Networks Voltage Regulation with Distribution Management System Coordinated Controller. *Electric Power Systems Research*, 78(4):634–645, April 2008. ISSN 03787796. doi: 10.1016/j.epsr.2007.05.007. URL `http://linkinghub.elsevier.com/retrieve/pii/S0378779607001125`.

Avrim L. Blum and Merrick L. Furst. Fast Planning Through Planning Graph Analysis. *Artificial Intelligence*, 90:281–300, 1997. ISSN 00043702. doi: 10.1016/S0004-3702(96)00047-1.

Blai Bonet and Héctor Geffner. Planning as Heuristic Search: New Results. *Recent Advances in AI Planning*, 1809:360–372, 2000. URL `http://link.springer.com/chapter/10.1007/10720246_28`.

Blai Bonet and Héctor Geffner. Planning as Heuristic Search. *Artificial Intelligence*, 129(February 2000):5–33, 2001. ISSN 00043702. doi: 10.1016/S0004-3702(01)00108-4.

Joshua Campion, Chris Dent, Maria Fox, Derek Long, and Daniele Magazzeni. Challenge: Modelling Unit Commitment as a Planning Problem. In *International Conference on Automated Planning and Scheduling*, pages 452–456, 2013. URL `http://www.aaai.org/ocs/index.php/ICAPS/ICAPS13/paper/download/6041/6215`.

Jianing Cao, Keith Bell, Amanda Coles, and Andrew Coles. Voltage Control of Distribution Network using an Artificial Intelligence Planning Method. In *21st International Conference on Electricity Distribution*, pages 6–9, 2011. URL `http://homepages.eee.strath.ac.uk/~kbell/Publications/CIREDpaper_JianingCao_1112.pdf`.

Michael Cashmore, Maria Fox, Tom Larkworthy, Derek Long, and Daniele Magazzeni. AUV Mission Control via Temporal Planning. In *Robotics and Automation (ICRA), 2014 IEEE International Conference on*, pages 6535–6541, 2014. ISBN 9781479936847. doi: 10.1109/ICRA.2014.6907823.

Antonio Ceballos, Saddek Bensalem, Amedeo Cesta, Lavindra De Silva, Simone Fratini, Fèlix Ingrand, J Ocon, Andrea Orlandini, Frèdèric Py, Kanna Rajan, Riccardo Rasconi, and van Michel Winnendael. A Goal-Oriented Autonomous Controlled for Space Exploration. In *ASTRA*, volume 11, 2011.

Yixn Chen, Benjamin W. Wah, and Chih-Wei Hsu. Temporal Planning using Subgoal Partitioning and Resolution in SGPlan. *Journal of Artificial Intelligence Research*, 26:323–369, 2006.

Carleton Coffrin and Pascal Van Hentenryck. A Linear-Programming Approximation of AC Power Flows. *CoRR*, abs/1206.3:1–13, 2012. URL `http://arxiv.org/abs/1206.3614`.

Amanda Coles, Andrew Coles, Maria Fox, and Derek Long. Extending the Use of Inference in Temporal Planning as Forwards Search. In *International Conference on Automated Planning and Scheduling*, pages 66–73, 2009a.

Amanda Coles, Andrew Coles, Maria Fox, and Derek Long. Forward-Chaining Partial-Order Planning. *International Conference on Automated Planning and Scheduling*, pages 42–49, 2010. URL `http://www.aaai.org/ocs/index.php/ICAPS/ICAPS10/paper/download/1421/1527http://www.aaai.org/ocs/index.php/ICAPS/ICAPS10/paper/viewFile/1421/1527/`.

Amanda Coles, Andrew Coles, Maria Fox, and Derek Long. COLIN: Planning with Continuous Linear Numeric Change. *Journal of Artificial Intelligence Research*, 44:1–96, 2012. ISSN 10769757. doi: 10.1613/jair.3608.

Andrew Coles, Maria Fox, Derek Long, and Amanda Smith. A Hybrid Relaxed Planning Graph-LP Heuristic for Numeric Planning Domains. In *International Conference on Automated Planning and Scheduling*, pages 52–59, 2008. ISBN 9781577353867.

Andrew Coles, Maria Fox, Keith Halsey, Derek Long, and Amanda Smith. Managing Concurrency in Temporal Planning using Planner-Scheduler Interaction. *Artificial Intelligence*, 173(1):1–44, 2009b. ISSN 00043702. doi: 10.1016/j.artint.2008.08.003. URL `http://dx.doi.org/10.1016/j.artint.2008.08.003`.

Andrew Coles, Amanda Coles, Maria Fox, and Derek Long. POPF2: a Forward-Chaining Partial Order Planner. In Dergio Jimenez Angela Garcìa-Olaya and Carlos Linares Lòpez, editors, *The Seventh International Planning Competition: Description of Participant Planners of the Deterministic Track*, pages 65–70. Planning and Learning Group (PLG) Universidad Carlos III de Madrid, 2011.

William Cushing, Subbarao Kambhampati, Mausam, and Daniel S. Weld. When is Temporal Planning Really Temporal? In *International Joint Conference on Artificial Intelligence*, pages 1852–1859, 2007.

Euan M. Davidson, Michael J. Dolan, Sephen D. J. McArthur, and Grant W. Ault. The Use of Constraint Programming for the Autonomous Management of Power Flows. In *15th International Conference on Intelligent System Applications to Power Systems*, pages 1–7, 2009.

Euan M. Davidson, Michael J. Dolan, Grant W. Ault, and Sephen D. J. McArthur. AuRA-NMS: An Autonomous Regional Active Network Management System for EDF Energy and SP Energy Networks. In *IEEE Power and Energy Society General Meeting*, pages 1–6, 2010.

Lavindra De Silva, Amit Kumar Pandey, Mamoun Gharbi, and Rachid Alami. Towards Combining HTN Planning and Geometric Task Planning. In *Combined Robot Motion Planning and AI Planning for Practical Applications Workshop*, pages 1–6, 2013.

Rina Dechter, Itay Meiri, and Judea Pearl. Temporal Constraint Networks. *Artificial Intelligence*, 49:61–95, 1991. ISSN 00043702. doi: 10.1016/0004-3702(91)90006-6.

Giuseppe Della Penna, Daniele Magazzeni, Fabio Mercorio, and Benedetto Intrigila. UPMurphi: A Tool for Universal Planning on PDDL+ Problems. In *International Conference on Automated Planning and Scheduling*, pages 106–113, 2009. URL `http://www.aaai.org/ocs/index.php/ICAPS/ICAPS09/paper/download/707/1100`.

Department for Business Enterprise & Regulatory Reform. UK Renewable Energy Strategy. Technical Report June, HM Government, 2008.

Minh B. Do and Subbarao Kambhampati. Sapa: A Scalable Multi-objective Heuristic Metric Temporal Planner. *Journal of Artificial Intelligence Research*, 20:155–194, 2003. URL `http://citeseerx.ist.psu.edu/viewdoc/summary;jsessionid=08DE533A6362CECD89C5FA14A30A724D?doi=10.1.1.12.2783`.

Christian Dornhege, Patrick Eyerich, and Thomas Keller. Semantic Attachments for Domain-Independent Planning Systems. In *International Conference on Automated Planning and Scheduling*, pages 114–121, 2009. URL `http://link.springer.com/chapter/10.1007/978-3-642-25116-0_9`.

Stefan Edelkamp. Planning with Pattern Databases. In *European Conference on Planning (ECP)*, pages 13–34, 2001.

Stefan Edelkamp. Taming Numbers and Duration in the Model Checking Integrated Planning System. *Journal of Artificial Intelligence Research*, 20:195–238, 2003.

Stefan Edelkamp and Shahid Jabbar. MIPS-XXL : Featuring External Shortest Path Search for Sequential Optimal Plans and External Branch-And-Bound for Optimal Net Benefit. In *Booklet for the 6th International Planning Competition*, 2008.

Patrick Eyerich, Robert Mattmüller, and Gabriele Röger. Using the Context-Enhanced Hdditive heuristic for Temporal and Numeric Planning. In *International Conference on Automated Planning and Scheduling*, pages 130–137, 2009. URL `http://link.springer.com/chapter/10.1007/978-3-642-25116-0_6`.

Bruce Fardanesh. Future Trends in Power Control. *IEEE Computer Applications in Power*, 15(3): 24–31, 2002.

Maria Fox and Derek Long. PDDL2. 1: An Extension to PDDL for Expressing Temporal Planning Domains. *Journal of Artificial Intelligence Research*, 20:61–124, 2003. URL http://www.aaai.org/Papers/JAIR/Vol20/JAIR-2002.pdf.

Maria Fox and Derek Long. Modelling Mixed Discrete-Continuous Domains for Planning. *Journal of Artificial Intelligence Research*, 27:235–297, 2006. ISSN 10769757. doi: 10.1613/jair.2044.

Maria Fox, Derek Long, and Keith Halsey. An Investigation into the Expressive Power of PDDL2.1. In *European Conference of Artificial Intelligence (ECAI)*, pages 328–342, 2004. URL http://books.google.co.uk/books?hl=en&lr=&id=rU_onmzozu0C&oi=fnd&pg=PA338&dq=derek+long&ots=w4oqbYT9S5&sig=3J3nREwWEFidESKJxZPl-04kbXw.

Maria Fox, Derek Long, and Daniele Magazzeni. Plan-based Policies for Efficient Multiple Battery Load Management. *Journal of Artificial Intelligence Research*, 44:335–382, 2012. URL http://dl.acm.org/citation.cfm?id=2387940.

Jeremy Frank and Ari Jónsson. Constraint-based Attribute and Interval Planning. *Journal of Constraints, Special Issue on Constraints and Planning*, 8:339–364, 2003. ISSN 13837133. doi: 10.1023/A:1025842019552.

Héctor Geffner and Patrik Haslum. Admissible Heuristics for Optimal Planning. In *Conference on Artificial Intelligence Planning and Scheduling*, pages 140–149, 2000. URL http://www.aaai.org/Papers/AIPS/2000/AIPS00-015.pdf.

Alfonso Gerevini, Alessandro Saetti, and Ivan Serina. An Approach to Temporal Planning and Scheduling in Domains with Predictable Exogenous Events. *Journal of Artificial Intelligence Research*, 25:187–231, 2006. URL http://www.aaai.org/Papers/JAIR/Vol25/JAIR-2506.pdf.

Alfonso E. Gerevini, Patrik Haslum, Derek Long, Alessandro Saetti, and Yannis Dimopoulos. Deterministic Planning in the 5th IPC: PDDL3 and Experimental Evaluation of the Planners. *Artificial Intelligence*, 173(5-6):619–668, 2009. URL http://scholar.google.co.uk/scholar?q=derek+long&hl=en&btnG=Search&as_sdt=2001&as_sdtp=on#79.

Government
Digital Service. UK Historical Electricity Data. https://www.gov.uk/government/statistical-data-sets/historical-electricity-data-1920-to-2011, 2014.

Peter Gregory, Derek Long, Maria Fox, and J. Christopher Beck. Planning Modulo Theories: Extending the Planning Paradigm. *International Conference on Automated Planning and Scheduling*, pages 65–73, 2012. URL http://www.aaai.org/ocs/index.php/ICAPS/ICAPS12/paper/download/4693/4715.

Peter E. Hart, Nils J. Nilsson, and Bertram Raphael. Formal Basis for the Heuristic Determination of Minimal Cost Path. *Systems Science and Cybernetics, IEEE Transactions on*, 4(2):100–107, 1968. doi: 10.1109/TSSC.1968.300136.

Patrik Haslum and Héctor Geffner. Heuristic Planning with Time and Resources. In *European Conference on Planning (EPC)*, 2001. URL https://www.ida.liu.se/divisions/aiics/publications/ECP-2001-Heuristic-Planning-Time.pdf.

Patrik Haslum, Adi Botea, Malte Helmert, Blai Bonet, and Sven Koenig. Domain-independent Construction of Pattern Database Heuristics for Cost-optimal Planning. In *American Association for Artificial Intelligence*, pages 1007–1012, 2007.

Malte Helmert. The Fast Downward Planning System. *Journal of Artificial Intelligence Research*, 26:191–246, 2006. ISSN 10769757. doi: 10.1613/jair.1705.

Jörg Hoffmann. The Metric-FF Planning System: Translating "Ignoring Delete Lists" to Numeric State Variables. *Journal of Artificial Intelligence Research*, 20:291–341, 2003. URL `http://www.aaai.org/Papers/JAIR/Vol20/JAIR-2011.pdf`.

Jörg Hoffmann and Stefan Edelkamp. The Deterministic Part of IPC-4: An Overview. *Journal of Artificial Intelligence Research*, 24:519–579, 2005.

Jörg Hoffmann and Bernhard Nebel. The FF Planning System: Fast Plan Generation Through Heuristic Search. *Journal of Artificial Intelligence Research*, 14(1):253–302, 2001. URL `http://arxiv.org/abs/1106.0675`.

Jörg Hoffmann, Julie Porteous, and Laura Sebastia. Ordered Landmarks in Planning. *Journal of Artificial Intelligence Research*, 22:215–278, 2004.

Zechun Hu and Furong Li. Cost-Benefit Analyses of Active Distribution Network Management, Part II: Investment Reduction Analysis. *IEEE Transactions on Smart Grid*, 3(3):1075–1081, 2012.

S Kameshwaran, Alfiya Tezabwala, Alain Chabrier, Julian Payne, and Fabio Tiozzo. Integrated Operations (Re-) Scheduling from Mine to Ship. In *International Conference on Automated Planning and Scheduling*, pages 416–424, 2013. URL `http://www.researchgate.net/profile/Kameshwaran_Sampath/publication/235686593_Integrated_Operations_(Re-)Scheduling_from_Mine_to_Ship/links/00b4951cc142a836ec000000.pdf`.

Bharat Ranjan Kavuluri and Senthil U. TILSapa - Timed Initial Literals Using SAPA. In *Booklet for the 4th International Planning Competition*, pages 46–47, 2004.

Emil Keyder, Jörg Hoffmann, and Patrik Haslum. Semi-relaxed plan heuristics. In *International Conference on Automated Planning and Scheduling*, pages 128–136, 2012. ISBN 9781577355687.

James E. Kings, Samuel C. E. Jupe, and Philip C. Taylor. Network State-Based Algorithm Selection for Power Flow Management Using Machine Learning. *IEEE Transactions on Power Systems*, PP(99):1–8, 2014.

Daniel L. Kovacs. BNF definition of PDDL 3.1 (completely corrected). In *Unpublished manuscript from the IPC-2011 website*, volume 1, pages 1–5, 2011.

Philippe Laborie and Malik Ghallab. Planning with Sharable Resource Constraints. In *International Joint Conference on Artificial Intelligence*, pages 1643–1649, 1995. URL `http://euroc.laas.fr/pub/ria/malik/articles/ijcai95.pdf`.

Fabien Lagriffoul, Dimitar Dimitrov, Julien Bidot, Alessandro Saffiotti, and Lars Karlsson. Efficiently Combining Task and Motion Planning Using Geometric Constraints. *International Journal of Robotics Research*, 33(14):1726–1747, 2013.

Mats Larsson. *Coordinated Voltage Control in Electric Power Systems*. PhD thesis, Lund University, 2000.

Johannes Löhr, Patrick Eyerich, Thomas Keller, and Bernhard Nebel. A Planning Based Framework for Controlling Hybrid Systems. In *International Conference on Automated Planning and Scheduling*, pages 164–171, 2012. URL `http://www.aaai.org/ocs/index.php/ICAPS/ICAPS12/paper/download/4708/4726`.

Johannes Löhr, Martin Wehrle, Maria Fox, and Bernhard Nebel. Symbolic Domain Predictive Control. In *American Association for Artificial Intelligence*, pages 2315–2321, 2014.

Jan Machowski, Janusz W. Bialek, and James R. Bumby. *Power System Dynamics: Stability and Control*. John Wiley & Sons, Ltd, second edition, 2008. ISBN 9780470725580.

Jan M. Maciejowski. *Predictive Control with Constraints*. Prentice Hall, Essex, England, 2002.

Sephen D. J. McArthur, Philip C. Taylor, Grant W. Ault, James E. Kings, Dimitrios Athanasiadis, Varvara Alimisis, and Maciej Czaplewski. The Autonomic Power System-Network Operation and Control Beyond Smart Grids. In *IEEE PES International Conference and Exhibition on Innovative Smart Grid Technologies*, pages 1–7, 2012. ISBN 9781467325974. URL `http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=6465807`.

Drew McDermott, Malik Ghallab, Adele Howe, Craig Knoblock, Ashwin Ram, Manuela Veloso, Daniel Weld, and David Wilkins. PDDL - The Planning Domain Definition Language. Technical report, Yale Center for Computational Vision and Control, 1998. URL `http://www.citeulike.org/group/13785/article/4097279`.

Mehryar Mohri, Afshin Rostamizadeh, and Ameet Talwalkar. *Foundations of Machine Learning*. The MIT Press, 2012.

Joseph Mutale. Benefits of Active Management of Distribution Networks with Distributed Generation. In *Power Systems Conference and Exposition*, pages 601–606, 2006. ISBN 1-4244-0177-1. doi: 10.1109/PSCE.2006.296385. URL `http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=4075824http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=4075824`.

National Grid PLC. National Grid UK. http://www.nationalgrid.com/uk/Electricity/Data/Demand+Data/, 2012. URL `http://www.nationalgrid.com/uk/Electricity/Data/Demand+Data/`.

Sergio Nùñez, Daniel Borrajo, and Carlos Linares López. MIPLAN. In *Eighth International Planning Competition, Planning and Learning Part*, 2014.

Masahiro Ono, Wesley Graybill, and Brian C. Williams. Risk-sensitive Plan Execution for Connected Sustainable Home. In *4th ACM Workshop on Embedded Sensing Systems for Energy-Efficiency in Buildings*, pages 45–52, New York, New York, USA, 2012. ACM Press. ISBN 9781450311700. doi: 10.1145/2422531.2422541. URL `http://dl.acm.org/citation.cfm?doid=2422531.2422541http://dl.acm.org/citation.cfm?id=2422541`.

Simon Parkinson, Peter Gregory, Andrew Longstaff, and Andrew Crampton. Automated Planning for Multi-Objective Machine Tool Calibration: Optimising Makespan and Measurement Uncertainty. In *International Conference on Automated Planning and Scheduling*, pages 421–429, 2014.

Judea Pearl. *Heuristics: Intelligent Search Strategies for Computer Problem Solving*. Addison-Wesley Longman Publishing Co., Inc., 1984. ISBN 0-201-05594-5.

Edwin P. D. Pednault. ADL: Exploring the Middle Ground Between STRIPS and the Situation Calculus. In *First International Conference on Principles of Knowledge Representation and Reasoning*, pages 324–332, 1989.

Malahat Peikherfeh, Hossein Seifi, and Kazem Sheikh-El-Eslami. Active Management of Distribution Networks in Presence of Distributed Generations. In *International Conference on Clean Electrical Power*, pages 725–729, 2011. ISBN 9781424489305. URL `http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=6036383`.

Dzung T. Phan and Jayant R. Kalagnanam. Some Efficient Optimization Methods for Solving the Security-Constrained Optimal Power Flow Problem. *IEEE Transactions on Power Systems*, 29 (2):863–872, 2014. doi: 10.1109/TPWRS.2013.2283175.

Chiara Piacentini, Varvara Alimisis, Maria Fox, and Derek Long. Combining a Temporal Planner with an External Solver for the Power Balancing Problem in an Electricity Network. In *International Conference on Automated Planning and Scheduling*, 2013. URL `http://www.aaai.org/ocs/index.php/ICAPS/ICAPS13/paper/download/6047/6209`.

Ira Pohl. Heuristic Search Viewed as Path Finding in a Graph. *Artificial Intelligence*, 1(3A): 193–204, 1970. ISSN 00043702. doi: 10.1016/0004-3702(70)90007-X.

Rene Prenc, Davor Škrlec, and Vitomir Komen. A Novel Load Flow Algorithm for Radial Distribution Networks with Dispersed Generation. *Tehnicki Vjesnik*, 20(6):969–977, 2013.

Frédéric Py, Kanna Rajan, and Conor McGann. A Systematic Agent Framework for Situated Autonomous Systems. *Autonomous Agents and Multiagent Systems*, pages 583–590, 2010. URL `http://dl.acm.org/citation.cfm?id=1838183`.

Silvia Richter, Malte Helmert, and Matthias Westphal. Landmarks Revisited. In *American Association for Artificial Intelligence*, pages 975–982, 2008. ISBN 9781577353683.

Vaughan Roberts, Alan Collinson, and Andrew Beddoes. Active Networks for the Accommodation of Dispersed Generation. In *17th International Conference on Electricity Distribution*, pages 12–15, 2003. URL `http://www.cired.net/publications/cired2003/reports/R4-51.pdf`.

Stuart J. Russell and Peter Norvig. *Artificial Intelligence: a Modern Approach*. Pearson Education, 2 edition, 2003. ISBN 0137903952. URL `https://www.cis.uab.edu/courses/cs760/Spring-660-2007/7A-660-PLUS-SYLLABUS-DRAFT.pdf`.

Erik Sandewall. *Features and Fluents (Vol. 1): The Representation of Knowledge About Dynamical Systems*. Oxford University Press, Inc., 1994.

Paul Scott and Sylvie Thiébaux. Dynamic Optimal Power Flow in Microgrids using the Alternating Direction Method of Multipliers. *CoRR*, pages 1–8, October 2014. URL `http://arxiv.org/abs/1410.7868`.

Jendrik Seipp, Silvan Sievers, and Frank Hutter. Fast Downward Cedalion. In *Eighth International Planning Competition, Planning and Learning Part*, 2014a.

Jendrik Seipp, Silvan Sievers, and Frank Hutter. Fast Downward SMAC. In *Eighth International Planning Competition, Planning and Learning Part*, 2014b.

David E. Smith and Daniel S. Weld. Temporal Planning with Mutual Exclusion Reasoning. In *International Joint Conference on Artificial Intelligence*, pages 326–337, 1999. URL `http://cs.tju.edu.cn/faculty/zyfeng/Course/AI/ISI/ijcai99-tgp.pdf`.

Siddharth Srivastava, Lorenzo Riano, Stuart Russell, and Pieter Abbeel. Using Classical Planners for Tasks with Continuous Operators in Robotics. In *ICAPS Workshop on Planning and Robotics*, pages 1–9, 2013.

Stationery Office. *The Electricity Safety, Quality and Continuity Regulations 2002*. Number 2665. 2002.

Brian Stott, Jorge Jardim, and Ongun Alsac. DC Power Flow Revisited. *IEEE Transactions on Power Systems*, 24(3):1290–1300, August 2009. ISSN 0885-8950. doi: 10.1109/TPWRS.2009.2021235. URL http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=4956966.

Ching-tzong Su and Chu-sheng Lee. Network Reconfiguration of Distribution Systems Using Improved Mixed-Integer Hybrid. *IEEE Transactions on Power Delivery*, 18(3):1022–1027, 2003.

Sylvie Thiébaux and Marie-Odile Cordier. Supply Restoration in Power Distribution Systems - a Benchmark for Planning Under Uncertainty. In *6th European Conference on Planning*, 2001. URL http://scalab.uc3m.es/~ecp01/ecp01.pdf#page=101.

Sylvie Thiébaux, Jörg Hoffmann, and Bernhard Nebel. In Defense of PDDL Axioms. *Artificial Intelligence*, 168(1-2):38–69, October 2005. ISSN 00043702. doi: 10.1016/j.artint.2005.05.004. URL http://linkinghub.elsevier.com/retrieve/pii/S0004370205000810.

Sylvie Thiébaux, Carleton Coffrin, Hassan Hijazi, and John Slaney. Planning with MIP for Supply Restoration in Power Distribution Systems. In *International Joint Conference on Artificial Intelligence*, pages 2900–2907, 2013. URL http://dl.acm.org/citation.cfm?id=2540546.

Henning Tischer and Gregor Verbic. Towards a Smart Home Energy Management System - A Dynamic Programming Approach. In *IEEE PES Innovative Smart Grid Technologies*, pages 1–7. IEEE, November 2011. ISBN 978-1-4577-0875-6. doi: 10.1109/ISGT-Asia.2011.6167090. URL http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6167090.

TNEI Services Ltd. IPSA Power. http://www.ipsa-power.com/, 2010.

University of Washington Electrical Engineering. Power Systems Test Case Archive. http://www.ee.washington.edu/research/pstca/, 2015. URL http://www.ee.washington.edu/research/pstca/.

Menkes van den Briel, J. Benton, Subbarao Kambhampati, and Thomas Vossen. An LP-Based Heuristic for Optimal Planning. In *Principles and Practice of Constraint Programming*, pages 651–665, 2007.

Vincent Vidal. CPT4: An Optimal Temporal Planner Lost in a Planning Competition without Optimal Temporal Track. In Dergio Jimenez Angela Garcìa-Olaya and Carlos Linares Lòpez, editors, *The Seventh International Planning Competition: Description of Participant Planners of the Deterministic Track*, pages 25–28. Planning and Learning Group (PLG) Universidad Carlos III de Madrid, 2011. URL http://www.plg.inf.uc3m.es/ipc2011-deterministic/attachments/Results/ipc2011-booklet.pdf#page=25.

Vincent Vidal and Héctor Geffner. Branching and Pruning: An Optimal Temporal POCL Planner Based on Constraint Programming. *Artificial Intelligence*, 170:298–335, 2006. ISSN 00043702. doi: 10.1016/j.artint.2005.08.004.

Marc Vilain and Henry Kautz. Constraint Propagation Algorithms for Temporal Reasoning The Interval Algebra. In *American Association for Artificial Intelligence*, volume 86, pages 377–382, 1986.

Jesùs Virseda, Daniel Borrajo, and Vidal Alcàzar. LLAMA: Learning LAMA. In *Eighth International Planning Competition, Planning and Learning Part*, 2014.

Richard W. Weyhrauch. Prolegomena to a Theory of Mechanized Formal Reasoning. *Artificial intelligence*, 13(1A):133–170, 1980.

Frank Wilcoxon. Individual Comparisons by Ranking Methods. In Samuel Kotz and Norman L. Johnson, editors, *Breakthroughs in Statistics*, pages 196–292. Springer New York, 1992.

Qiang Yang, Javier A. Barria, and Tim C. Green. Communication Infrastructures for Distributed Control of Power Distribution Networks. *IEEE Transactions on Industrial Informatics*, 7(2): 316–327, 2011. URL http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=5746548.

Tjalling J. Ypma. Historical Development of the Newton-Raphson Method. *SIAM review*, 35 (4):531–551, 1995. URL http://www.jstor.org/stable/2132904http://epubs.siam.org/doi/pdf/10.1137/1037125.

Junhui Zhao, Caisheng Wang, Bo Zhao, Feng Lin, Quan Zhou, and Yang Wang. A Review of Active Management for Distribution Networks: Current Status and Future Development Trends. *Electric Power Components and Systems*, 42(3-4):280–293, March 2014. ISSN 1532-5008. doi: 10.1080/15325008.2013.862325. URL http://www.tandfonline.com/doi/abs/10.1080/15325008.2013.862325.

Ray Daniel Zimmerman, Carlos Edmundo Murillo-Sánchez, and Robert Jhon Thomas. MATPOWER: Steady-State Operations, Planning, and Analysis Tools for Power Systems Research and Education. *IEEE Transactions on Power Systems*, 26(1):12–19, 2011. ISSN 0885-8950. doi: 10.1109/TPWRS.2010.2051168. URL http://ieeexplore.ieee.org/ielx5/59/5695072/05491276.pdf?tp=&arnumber=5491276&isnumber=5695072.