Short Term Unit Commitment as a Planning Problem

JOSHUA ROBERT CAMPION

Short Term Unit Commitment as a Planning Problem

Joshua Campion

A Thesis presented for the degree of
Master of Philosophy

Power Systems Engineering Group
Department of Engineering and Computer Sciences
University of Durham
England

February 2014

Short Term Unit Commitment as a Planning Problem

Abstract

Josh Campion

'Unit Commitment', setting online schedules for generating units in a power system to ensure supply meets demand, is integral to the secure, efficient, and economic daily operation of a power system. Conflicting desires for security of supply at minimum cost complicate this. Sustained research has produced methodologies within a guaranteed bound of optimality, given sufficient computing time.

Regulatory requirements to reduce emissions in modern power systems have necessitated increased renewable generation, whose output cannot be directly controlled, increasing complex uncertainties. Traditional methods are thus less efficient, generating more costly schedules or requiring impractical increases in solution time.

Meta-Heuristic approaches are studied to identify why this large body of work has had little industrial impact despite continued academic interest over many years. A discussion of lessons learned is given, and should be of interest to researchers presenting new Unit Commitment approaches, such as a Planning implementation.

Automated Planning is a sub-field of Artificial Intelligence, where a timestamped sequence of predefined actions manipulating a system towards a goal configuration is sought. This differs from previous Unit Commitment formulations found in the literature. There are fewer times when a unit's online status switches, representing a Planning action, than free variables in a traditional formulation. Efficient reasoning about these actions could reduce solution time, enabling Planning to tackle Unit Commitment problems with high levels of renewable generation.

Existing Planning formulations for Unit Commitment have not been found. A successful formulation enumerating open challenges would constitute a good benchmark problem for the field. Thus, two models are presented. The first demonstrates the approach's strength in temporal reasoning over numeric optimisation. The second balances this but current algorithms cannot handle it. Extensions to an existing algorithm are proposed alongside a discussion of immediate challenges and possible solutions. This is intended to form a base from which a successful methodology can be developed.

Declaration

The work in this thesis is based on research carried out in the School of Engineering and Computer Sciences, Durham University, England. No part of this thesis has been submitted elsewhere for any other degree or qualification and it is all my own work unless referenced to the contrary in the text.

Acknowledgements

Firstly, I would like to thank my supervisor Chris Dent for all his support, help and direction throughout my time as a post graduate student, without which I could not have seen this project through to completion. Secondly, Janusz Bialek for the numerous discussions and feedback on the content and direction of this project. I am very grateful to all in the Durham University Engineering and Computer Sciences department, Durham Energy Institute Center for Doctoral Training, and Autonomic Power Systems Project for discussions on everything Power Systems and Energy related.

I am very grateful to have been able to collaborate with the Planning Group at King's College London, and would like to thank them for all their help and discussions with regards to everything Planning. For everything from providing code and data not publicly available to many visits and meetings, I am grateful to have been able to work as part of your group.

I would like to express particular gratitude to Maria Fox and Derek Long for all their help and guidance with Automated Planning, and the publication and presentation of this work in ICAPS 2013. I could not have studied this area without all of their help, advice and guidance. And, to Daniele Magazzeni of the same group who has spent much time helping with me all issues including modelling in PDDL, constructing and running the UPMURPHI model, refining the ideas of the Separated Model for Unit Commitment and all his work on the ICAPS 2013 paper. I am extremely grateful for all the hours put in on my behalf.

Finally to anybody who attempts to develop the ideas presented here, I hope they can be of use.

Contents

List of Figures

Chapter 1

Introduction

1.1 What Is Unit Commitment?

A power system consists of many power stations, each containing a single or multiple generating units. There are a few hundred units in the UK network [1], and thousands in some US networks [2]. These units, or subsets of these units, are controlled centrally by a System Operator (SO). Whilst some types of generating units such as hydro power can come online and begin supplying power to the grid in minutes, others such as coal and nuclear take many hours to warm up before they can begin feeding power into the network [3, Chp 2]. Because of this, system operators and utilities companies must know well in advance when to bring a generating unit online.

Unit Commitment is an optimisation problem, specifically, scheduling which generating units will be online at what times, in order to meet electricity demand over a given horizon subject to the many physical constraints on a power system [3, Chp 5]. The problem can vary in many ways, by including different subsets of operating constraints, some minimise the cost of generation, others optimise profit in a competitive market.

Economic Dispatch is the name for the intrinsically linked sub-problem of the exact output for each generating unit, for each time period that unit is online [3, §3.1]. Again there are many constraints and variants. A version of Economic Dispatch is performed as part of Unit Commitment to assess the cost of a specific scheduling of

units.

Even though Economic Dispatch is calculated for the full planning horizon in Unit Commitment, final unit outputs are not set by the values in the Unit Commitment solution but by more complex Economic Dispatch algorithms run closer to real time. This is because Unit Commitment is a very complex optimisation problem and solving with full complexity would be infeasible given current technology. Also final power outputs at time of delivery should be based on more up to date forecasts of customer demand. This means the key part of the solution for Unit Commitment is not the outputs of the generators, but the online schedules produced for each unit. In principle one could implement Rolling Unit Commitment, where Unit Commitment is recalculated at regular intervals throughout the time horizon, increasing the accuracy of the outputs produced. In practice market reforms would be required in many operating areas for this to take place, and such action takes much time to implement. The current market schedules relating to Unit Commitment for some US operating areas can be found in [4].

Unit Commitment is an extremely important problem to the power industry, one that is integral to the secure, efficient, and economic daily operation of a power system. The operational cost of running a power system over just one day is huge, and with rising oil and gas prices the cost is ever increasing. Coupling this with customer, political, and regulatory, demands for lower prices has lead the industry towards an obsessive need to optimise every aspect of the system. Committing one portfolio of units over another could have very expensive consequences for the operational cost of that day, so ensuring the correct portfolio is chosen is imperative.

Mathematical Optimisation is a huge field of research, and consequently Unit Commitment implementations have become more and more sophisticated. Nevertheless, industrial applications still require parallel computing on clusters of machines to produce acceptably accurate results within practical time frames [5]. Early implementations could not claim optimality but the past 15 years has seen a huge uptake in Mixed Integer Programming (MIP) implementations for Unit Commitment solved using Branch and Bound[1], able to find a solution within a guaranteed

gap to optimality given sufficient time.

A mathematical optimisation problem specifies an objective function, the value of which is to be maximised or minimised over a set of decision variables given a collection of constraints imposed on the value of those decision variables. In a MIP, some of those variables take continuous values, others are restricted to take only integer values.

Much research has lead to fast solution algorithms for certain kinds of optimisation problems, where the objective function is linear or Quadratic in all decision variables, known as Linear Programming (LP) or Quadratic Programming (QP) problems. The simplex algorithm for LPs and Interior Point Methods for QPs are both efficient algorithms [6, Chp 3, 4] .

The Branch and Bound solution process involves repeatedly solving similar LPs or QPs throughout search. A MIP with either a linear or quadratic objective function, known as a MILP or MIQP, can make use of the above algorithms to solve their subproblems. The simplex algorithm can be 'warm started', meaning a solution which is close to optimal can be provided dramatically reducing search time to the true optimal. Interior Point Methods do not see such a dramatic reduction when warm starting. MILPs exploiting Simplex therefore gain a solution time advantage over MIQPs. For this reason much of the literature on Unit Commitment focuses on linear formulations, as shall be the case throughout this document.

To express Unit Commitment as a MILP it is necessary to discretise the time horizon, typically into 30 or 60 minute intervals. For each generating unit in the system, and each time period in the horizon, there is a binary decision variable representing whether that generating unit is online, and a real valued decision variable representing the proposed output of the generating unit. A proposed output is necessary to calculate a prospective cost.[2]

[1]Branch and Bound is the name of an algorithm for solving optimisation problems. Discussed in further detail in Section 2.2 it is often used synonymously with Branch and Cut in the literature, although there are technical differences between the two, discussed on p23.

[2]See Appendix A.1 for a description of a basic MIP implementation created for test purposes and solved with a commercially available solver.

The assumption that a unit's power output and total customer demand is constant over each period and changes instantaneously, is made. This is far from reality but as mentioned earlier, the exact power outputs of units are not set by Unit Commitment, only the online and offline schedules are set here[3], and this approximation is realistic enough that it is unlikely to affect the online schedules for each unit, and significantly reduces the complexity of the problem.

Whilst industrial applications for the past decade have focussed on solving the MIP formulation of Unit Commitment with Branch and Bound, there have been many other methods proposed in the academic literature. Mostly meta-heuristic methods, these can have fast solution times but lack the bound to optimality offered by Branch and Bound. Consequently the work has had little impact on industrial practices, as discussed in depth in Chapter 3.

There has been much success within academia and industry tackling many aspects of Unit Commitment, but open problems remain. Looking to the future, there are more factors affecting unit commitment than ever before. In the next 10-30 years large rises in variable renewable generation (VG) such as wind and solar farms is expected [7, §2.6], necessitated by regulatory commitments to reduce carbon emissions. These are "non-controllable" generators, providing electricity, the amount and timing of which is stochastic and beyond the operator's control.

Increased uptake of smart appliances, electric cars and other so called, "controllable loads" is also expected [8]. These are customer demands for electricity, but where the SO has some control over when the load is to be served, allowing more efficient use of VG. An SO can wait until variable generation outputs are realised, then serve controllable loads with excess renewable generation rather than serving them immediately with expensive thermal generation.

The uncertainty in time of availability, and scale of availability, of VG and con-

[3]Different networks have different market structures and Unit Commitment may be also be re-calculated closer to real time meaning that some schedules may change from the initial calculation. As many networks have quite different market structures most market specific optimisation criteria are tackled solely by the network operator and so are not widely considered in the literature and in academia as it would have a very restricted field of application.

trollable loads bring increased uncertainty to a problem which previously had very little, as demand patterns have always been very well understood when aggregated out over a large network. This extra uncertainty is in the net demand, and also the temporal aspects of the problem. These changes have a large impact on the Unit Commitment problem and are discussed further in Chapter 2.4, and motivate study into more flexible solution methodologies.

1.2 What is Automated Planning?

Automated Planning, also interchangeably referred to as 'AI Planning' and simply 'Planning', is concerned with sequencing actions to achieve goals. It therefore lends itself well to problems where the actions achieving the goal, the order of the actions, or some associated cost of actions are important.

More formally it is a field of Informatics concerned with generating action sequences to transform a system from an initial state to a goal state. Originally systems could consist solely of true / false facts and a state was a specific configuration of those facts. An action could manipulate the system by altering those true / false facts, and could be applied to a given state if and only if a set of 'preconditions' held true.

A problem would be to specify an initial state and some desired goal facts. A solution to a planning problem would consist of a sequence of actions, the cumulative effect of which would alter the facts in the initial state such that all goal facts became true.

Since its conception as a field many advances have been made. An ability to express types of objects allowed for a more natural expression of planning problems. Facts could now be global or associated with a given object and actions could have restricted applicability to only certain types of objects. Problem instances could be expressed in terms of multiple objects, with their properties modelled as facts.

Further advances to planning, such as the ability to encode numerical quantities into a state and encode temporal information such as action durations and exogenous events, enable problem descriptions to become very rich and realistic.

This description now is analogous to intuitive descriptions of real-world problems. Consider the problem of tasking a fleet of vehicles to deliver a collection of packages from source locations to destination locations. The domain might consist of vehicle objects, perhaps multiple types of vehicles each with different properties such as speed and cost, package objects perhaps of different size so only certain vehicles can transport certain combinations of packages, locations with relative distances and connectivity to other locations. Actions would involve loading vehicle with packages and moving vehicles between locations. For real-world problems such as this a planning formulation becomes a very natural one to express. Expressing this as a MIP for example, would be rather convoluted and unintuitive.

A planning formulation for Unit Commitment would have an object representing each unit, the actions would be bringing a unit online or offline, and the output values of each unit can be encoded into the world state. This model appears more intuitive than a MIP model, and as planning has shown strong performance on temporal problems it may be an interesting and novel approach to the problem. A more detailed introduction to Automated Planning can be found in Chapter 4.1. Discussions of the applicability of planning to Unit Commitment as well as planning model proposals can be found in Chapters 5 and 6.

1.3 Motivation for Study

As the power systems continues to evolve so too must Unit Commitment, and current solution methods in industry and academia have shortcomings when faced with these challenges.

A MIP formulation has the weakness that accurately representing the customer demand, especially once wind power integration is considered, requires a reasonably fine discretisation of time. Contrastingly, time constraints on the minimum online and offline duration of a unit mean that some units must be committed for a whole day, or most of a day, at a time so their temporal scheduling is already fixed, whilst even peaking units will only come online at most two or three times a day. This means that most of the binary decision variables for a generating unit are redundant

once one decision has been made early in the planning horizon. Reasoning about those excess variables unnecessarily extends a branch and cut solution process.

The difficulty for many meta-heuristic methods proposed in the literature is randomly generating feasible solutions to begin or iterate the search process. Most meta-heuristics contain a stochastic element which determines the next set of candidate solutions to use in the search process. The complexity of the constraints in a Unit Commitment problem however do not lend themselves well to this approach as random changes to the integer decision variables will more often than not result in an online schedule which violates the minimum online and offline physical constraints of a generating unit, which typically encompass multiple decision variables.

The time spent generating infeasible solutions, and reasoning that they are infeasible, extends the solution time and reduces the benefit of a fast meta-heuristic method over Branch and Bound, especially when the extra cost incurred through being further from optimality is considered. To overcome this many methods impose soft constraints, and penalise but do not disallow these infeasible solutions. Whilst this aids the solution process it is not representative of the definitive constraints necessitated by the Unit Commitment problem.

These existing methods come under more serious threats when considering the difficulties introduced by the future power systems discussed earlier. The problem becomes harder to model as a MIP, with stochastic formulations often intractable where Branch and Bound solution times extend beyond what is practical. The claimed faster solution times from Meta-Heuristic methods become more desirable in a stochastic setting where multiple runs are required, however as discussed in Chapter 3, it appears that a lack of consistent empirical evidence of this, no bounds to optimality and the difficulties in handling certain constraints are still off-putting to industry.

Whilst brute force and increased computing power will find a solution, there is a clear opportunity for a more flexible, scalable, and elegant solution methodology tailored for a problem with such temporal complexity and uncertainty.

As mentioned above, an LP can be solved very quickly, and so it is the integer aspect of Unit Commitment as a MILP which drastically increases the solution

time. The integer aspect of the problem is the online scheduling, for which a large number of binary variables are required for relatively few changes of online status. Determining which generating units to bring online at what times is perhaps more naturally expressed as a temporal planning problem. Most Unit Commitment schedules can be expressed with far fewer actions than binary decision variables required in a MILP.

With regards to meta-heuristic weaknesses, the complex temporal constraints that meta-heuristics have difficulty handling, are replaced with 2 simple actions in a Planning Model, from which the constraints are guaranteed to be enforced. This definitive removal of infeasible solutions is necessary for a solution methodology to be a consideration for practical implementation in industry.

Despite a large and sustained amount of academic interest for Unit Commitment with algorithms other than Branch and Bound there has been little impact on industrial practices. When proposing an alternative solution method to such an important industrial application it is necessary to assess and understand why previous practices were dismissed and current practices taken up. The field of Unit Commitment with Meta-Heuristics is so large and varied but with little industrial impact that its study, and a reflection upon contribution, is instructive to those pursuing new methodologies, to researchers aiming to further methodologies towards industrial uptake and to surveyors interested in the chronological development of Unit Commitment.

Finally there is a clear separation of the temporal and numeric aspects of the problem which could be exploited effectively using Automated Planning. Planners have been developed to perform heuristic search based on relaxed (less complex) variants of the main problem being solved. A natural relaxation of Unit Commitment is to fix subsets of the binary decision variables and solve the resulting LP, indeed this is an integral part of the branch and cut solution process.

This relaxation is a natural fit for a planning solver, which can reason effectively about temporal problems but less so about numerical optimisation. Supposing a planner were to perform reasoning about the temporal online schedules and receive guidance on optimisation from an external LP solver, each sub-problem would be

handled by a solution methodology well suited to the type of sub-problem being solved. Another PhD project in the same consortium project funding this research is demonstrating the effectiveness of planners incorporating external solvers into the solution process [9].

1.4 Contributions

The contributions of this thesis are four fold:

- A critical review of the literature solving Short Term Thermal Unit Commitment with Meta-Heuristic Algorithms discussing short comings of the research and possible reasons for the lack of industry uptake is presented in Chapter 3.
- A formulation of Unit Commitment as a planning problem which could be solved by an 'off-the-shelf' planner (i.e. without specific modifications) that supports the appropriate language constructs is presented in Chapter 5.
- A formulation of Unit Commitment as a planning problem exploiting the separation between numeric and temporal sub-problems discussed is presented in Chapter 6, detailing why this model may be advantageous.
- A discussion of extensions to an existing planner which may enable the separated model to be solved is given in Chapter 6.4.

The work on solving Unit Commitment is vast, and whilst there are other surveys in the literature ([10–13]) a more critical appraisal on Meta-Heuristics can provide a valuable point of discussion. With industrial focus solely on Branch and Bound, it is not only important to recognise the fact that the work appears to have had little industrial impact, but also to discuss why that might be the case.

The work discussed is evaluated on the realism of the model proposed, the breadth of test cases presented, comparison to other work, and any discussion of how to model further complications not handled. Of particular importance are the constraints respected and model size. Branch and Bound algorithms find optimal solutions quickly for smaller models with a representative set of constraints, so a

contribution claiming short solution times should be verified on models of the scale Branch and Bound would stall, not test systems alone.

Planning differs from all existing Unit Commitment methods the author is aware of in its formulation of problems using actions, not being based on a MIP. This reformulation into states and actions can be beneficial to Unit Commitment which typically suffers from problems of dimensionality. Much planning research has focussed on satisficing algorithms to quickly find feasible solutions. These can be guided by metrics but aren't designed with optimisation in mind.

The first planning model presented, which can be run on an appropriate existing planner, demonstrates this fallibility. A discussion of the performance on very small instances confirms the planner has poor optimisation performance, stalling on very small instances.

The author is unaware of this formulation being presented in the literature before and due to the difficult nature of expressing complex real-world problems as planning problems, this represents a contribution to the planning community. It is proposed to be a model from which an efficient solution algorithm for Unit Commitment could be developed, and highlights an opening in the existing planning literature.

Motivated by the specific problems of the first model, full details of the separated planning model and given. As this cannot be correctly handled by an existing planner, possible extensions to an existing planner are discussed which may result in the performant and efficient solution of Unit Commitment. This contribution is again of more value within the planning, where it may provide a starting point for discussions of how to efficiently tackle a whole class of problems similar in nature to Unit Commitment.

These contributions are more directly in the field of Automated Planning than in Power Systems Engineering. However planning represents a modern field of problem solving with much less time to develop than mathematical programming. Unit Commitment can be used as a benchmark problem in the field from which methods to tackle many advanced problems in Power Systems Engineering could be developed. Indeed there have been examples in the literature of planning tackling other Power Systems problems [9, 14, 15], and it is hoped that interest in the problem will lead

to advancements in both fields.

1.5 Structure

This thesis is structured as follows. Chapter 2 presents an Introduction to Unit Commitment in more detail, beginning with a brief survey of the chronological development of Unit Commitment and a detailed discussion of the emergence of MIP with Branch and Bound in industry. The chapter concludes with a discussion of challenges facing Unit Commitment model formulations and solvers looking forward.

Chapter 3 contains the aforementioned critical review of meta-heuristic methods. Chapter 4 begins with an introductory overview of the core concepts in Automated Planning necessary for the discussion of planning models throughout the remainder of the thesis. Following the introduction is a detailed discussion of the work in the literature focussed on planning for optimisation. Highlighted are the indications that planning may successfully tackle Unit Commitment, and that it is an excellent problem to strengthen the development of planning in a direction which has typically received less interest within the planning community.

Chapters 5 and 6 present the two aforementioned models concluding with a discussion of the extensions required to enable an existing planner to tackle this problem. Chapter 7 concludes by summarising the key outcomes from each contribution. The appendices present full expositions of model formulations and algorithms discussed, but not detailed, in the text.

Chapter 2

Unit Commitment

There has been much work in the literature regarding Unit Commitment. Being an integral part of power system operation over many decades, this problem has regularly featured as a demonstration of the capabilities of new algorithms (discussed in Chapter 3).

Whilst academically there has been a diversification in the number of different methods for solving Unit Commitment a full review of the field should also consider industry practices. This can be difficult as individual companies and system operators may be reluctant to give information which would jeopardise their intellectual property. Instead, events such as FERC technical conferences [16,17], bring together experts from academia and industry, from which an overview of current practices can be drawn.

As discussed in Chapter 1, Unit Commitment is changing. Increased variable generation and controllable loads bring increased uncertainty to a problem which previously had very little, as demand patterns have always been very well understood when aggregated out over a large network. The uncertainty in time of availability, and scale of availability, of variable generation is a large challenge to SOs who must accommodate accordingly. Generation not realising forecasted levels risks a drop in security and potential loss of load. Unexpected high levels of Variable Generation could lead to high curtailment of the renewables necessary to achieve a low carbon network.

A formulation including high levels of variability is known as Stochastic Unit

Commitment. A successful formulation of this problem, which can scale to real-world applications of RTOs has not yet been successfully demonstrated. With regulatory commitments to reduce emissions and incorporate renewables, a successful implementation is a key focus within the community.

This chapter presents a more in-depth introduction to the problem, then surveys the advances in Unit Commitment from initial implementations, to state-of-the-art industry practices. Methods for tackling upcoming changes to Unit Commitment and the power system more generally are also discussed.

2.1 A Unit Commitment Problem

It is not possible to define Unit Commitment by a single problem instance. In essence, as discussed in Chapter 1, the problem is to determine which generating units in a power system to take online or offline, at what times, to ensure supply meets predicted demand. Due to the very high costs involved in power generation this should be done at minimum cost. In practice the problem is constrained by the many physical complications that surround running a power system.

Different generating units have very different and complex physical characteristics, to model them all would make the problem very complex. Furthermore, the power grid and its topology imposes more physical constraints on the system due to the physical laws of electricity flow. To model the entire grid topology and all of its physical constraints would render even a system with a low number of units intractable. Thus, simplifications are made to the problem when it is being used in various situations.

For academic purposes running with relatively low computing power, many complications are ignored allowing for illustrative examples to highly certain features of a given (often implementation specific) model formulation or algorithm. In practice, system operators will have much more computing power available and thus be able to handle more complex constraints. Computing power is not the sole factor determining the level of realism to consider. Modern networks continue to increase in size and complexity and so the increases in computing power continue to be met

by increased problem complexity.

The time until the decisions of the unit commitment schedule are implemented is an important factor in deciding which constraints are worth modelling. In day-ahead Unit Commitment (considered throughout this project), where the schedule is being calculated 12-48 hours ahead of the horizon being considered, the forecasts for what demand will be are not 100% accurate. As such, calculating the power flows throughout the network (which are dependent on the places of generation and load not just amount of generation and load) would be an unnecessary complication. In a situation where the decision of which units to bring online are being made much closer to the times when the unit will come online, network considerations become much more important.

Suppose the operator needs to bring online more generation that can ramp up quickly as demand is higher than anticipated during the morning ramp. If most of the demand is centred around London, in the south of the network, the losses incurred from utilising extra generation in the north of the network might make such units less desirable than units in the south. In a situation such as this where the predicted demand levels are much more certain than in a day-ahead situation, network topology is much more important to consider.

Another factor in determining how realistic a problem can be tackled is the size of the system being considered. In the day-ahead scenario, most generating units in the system would be available to come online or offline at some point during the period of consideration. This means the number of units to consider is very high, increasing the combinatorial complexity of the problem. As such constraints such as network topology and advanced unit characteristics make the problem intractable due to its sheer size.

In the case above where network topology is important, the time period under consideration is likely to be much shorter, reducing the number of units which are able to respond in the necessary time frame. Fewer units reduces the combinatorial complexity and thus allows the specification of more complex constraints before the problem becomes intractable.

As mentioned previously, Economic Dispatch is the problem of determining the

amount of generation provided by each generating unit for each point in time. In a day ahead scenario, the forecasted demand is subject to significant changes which would alter the results of Economic Dispatch, were it to be run much closer to time of dispatch (real time). Also, when calculating the exact outputs of each unit, complications introduced by network topology should be considered. It is not simply losses but also problems such as real and reactive power balances and frequency control which play a major role in power system security on a minute by minute time scales. As such, despite Economic Dispatch being calculated as part of Unit Commitment, it is not these values that are used. More complex algorithms are run closer to real time.

The Economic Dispatch used as part of Unit Commitment is simply to provide guidance on which scheduling of units should be most economical given predicted demand. Performing Economic Dispatch with all possible constraints would be needlessly complicated and would render the problem intractable. There would be no additional advantage in the day ahead setting, when there is still time to perform such complex calculations closer to real time when demand is more accurately known and problem is smaller (as fewer units are able to respond on such a time frame, as discussed above).

For the above reasons, a single Unit Commitment problem definition cannot be given. It is beyond the scope of this project to discuss any and all such constraints which could be included as part of a the Unit Commitment problem. As an introduction for readers not familiar with Power Systems Engineering, an overview of typical features in a day-ahead time frame which is complex enough to represent an academic challenge but is not so complex to be intractable is given below. The interested reader should refer to more specialised resources for more detailed discussions of the many situation specific problem requirements.

A Typical Problem Specification

Suppose a planning horizon of 24 hours is to be considered. Then, for each generating unit in the system it should be decided over which intervals each unit is to be online. In order to demonstrate the economic benefit of a given scheduling, proposed outputs

for each unit should be given for each online period.

There are costs associated with starting units up, others incurred for the unit being online and marginal costs related to the output of the unit. Another cost to be included could be the value of lost load, where demand could be allowed to exceed supply but at a, typically very high, cost. Modern formulations can also take into account various renewable energy incentives, green house emission taxes and the possibility of network storage, where electricity can be over generated and stored to be deployed at very short notice at later times.

One objective is to minimise the total of these costs over the whole planning horizon. Other objects could consider generator bids for the prices of generation at certain levels of generation. Such a formulation is typically specific to a certain system as market strutters often differ between operating areas.

There are many constraints needed to ensure the model matches reality reasonably well. These include things such as:

- The minimum time a unit must remain online or offline for once it comes online or offline.
- An ability to bring a unit back online or offline for quicker, but at a different cost dependent on the time since the unit was last online or offline.
- The minimum and maximum possible outputs of a unit.
- The amount by which the output of a unit can vary over a given length of time, known as the ramp rate.
- Any pre-determined periods of availability for a given unit, such as for scheduled maintenance or as a result of decisions from previous planning periods.
- Any spinning reserve requirements[1].
- In certain circumstances, any network topology constraints.

[1]Spinning reserves are requirements for the system to be able to increase or decrease its combined output at any given time in the planning horizon to allow it to respond quickly to any

2.2 Classical Unit Commitment

Unit Commitment was being tackled as early as the 1960s [18], with attempts to assess hour-by-hour generation for a whole power system, rather than minimising the cost of running groups of units, as had been seen in prior work. Ranking the units according to a priority function, and dispatching according to the rule 'Switch off a generating unit if it will not be needed in the next x hours for serving demand or providing spinning reserve'. The ranking of these units was known as a Priority List.

The difficulty of this approach is prioritising the units to give accurate solutions, as well as determining the set of 'x hours' associated with each unit. The necessity to re-prioritise the units for different parts of the day also posed difficulties, i.e. high demand vs low demand, and to account for seasonable variabilities. This method was highly heuristic, but could be solved with the limited computing power available during early work on Unit Commitment.

Since the original demonstrations of speed and solution quality from Priority Lists, standalone and combined with multiple heuristics, a vast amount of Meta-Heuristic methods for solving Unit Commitment have been presented. These are considered extensively in Chapter 3.

Dynamic Programming approaches were another early method for tackling Unit Commitment [19, 20]. This Mathematical Programming approach can theoretically reach an optimal solution but suffers from the 'Curse of Dimensionality'[2]in practical applications. In a basic forward Dynamic Programming approach to Unit Commitment each possible combination of units being on or off for a time period is considered a state, and the least cost state is expanded to the next time period where each possible combination at that time period is considered and the least cost expanded. This continues until the problem is solved. Certainly at time of

unforeseen circumstances. This is typically achieved by running certain fast response units below their maximum output and above their minimum so they have spare capacity to quickly ramp up or down. The spinning reserve can be a fixed amount or dependent on the demand. The level of reserves is typically at the discretion of the system operator.

publication, and even with modern computing power, this approach quickly runs into memory issues. Backwards Dynamic Programming is also possible.

To address this, heuristics are used to guide the search and various approaches to state space reduction have been presented. Assuming these heuristic and reduction techniques are reasonable, guaranteed optimality is lost but the gap to it will hopefully be small.

State space reduction techniques have been numerous. Methods include:

- decomposing the problem in smaller, more manageable sub-problems [21]
- grouping units to reduce the combinatorial element of the search [22]
- truncating the space expanded at each step, only considering switching on those units within a certain portion of a Priority List [23, 24].

Other hybrid approaches were developed in an attempt to counteract the weaknesses of one method with the strengths of another. [25] demonstrated an iterative approach to generating a Priority List by re-prioritising the list of units based on their contribution to previous solutions. This brought increased computing requirements over static Priority Lists but reduced requirements over full Dynamic Programming, as the problem was simpler. The hybrid system was a success over both previous methods, producing cheaper schedules then Dynamic Programming alone an achieved up to 30 times speed up.

Lagrangian Relaxation was another popular method for solving Unit Commitment when computing power was limited [26]. In Lagrangian Relaxation (see for example [27]) the constraints from a linear programming problem are separated. One group of the constraints is incorporated into the objective function as a penalty, with each assigned a weight, known as a Lagrange Multiplier. This reduces the number of constraints on the problem and produces the relaxed variant. Solving the re-

[2]When a problem is solved by enumeration of feasible scenarios or solutions, the number of such scenarios or solutions increases, often exponentially, with the number of variables. A high number of variables results in too many scenarios or solutions to enumerate in a practical time frame. This is known as the 'Curse of Dimensionality'.

laxed variant can provide a lower (upper) bound on the optimal value of the original minimisation (maximisation) problem.

A related problem variant, known as the Lagrangian Dual of the problem can also be tackled. The highest value of the relaxed problem comes from the values of decision variables coupled with the lowest feasible values of the lagrange multipliers. Iteratively solving minimisation with respect to the lagrange multipliers with feasible values for the decision variables allows the solution of the relaxed problem (a lower bound on optimality) to converge towards true optimality.[3]

Lagrangian Relaxation requires more computing power than earlier methods but can produce higher quality solutions in practical situations [29]. Typically it is the hardest constraints in an LP, such as those linking decision variables, which are separated from the problem and become penalty terms in the objective function. In Unit Commitment the relaxed constraints are typically demand and reserve [29–32] removing the hard constraints which couple units.

The increased accuracy of Lagrangian Relaxation came in part from its systematic approach, iterating towards a global optimum based on a provable upper bound on gap to optimality. Figure 2.1 illustrates this. The realised solution to the Lagrangian Dual converges towards an optimum for the Lagrangian Dual problem. From this solution a feasible solution can be generated. Assuming a good generation scheme the feasible solution converges towards the optimum of the original problem as the dual problem converges to its optimum [29].

Complications to overcome when implementing Lagrangian Relaxation include initial values for the Lagrange multipliers, the scheme to update the lagrane multipliers for each iteration, and generating a feasible, integer, solution from the relaxed, continuous, dual solution [33]. Another difficulty when implementing Lagrangian Relaxation for Unit Commitment is its lack of flexibility. Adding new constraints to the model requires recalculation of the dual problem. In turn, this can necessitate changes to the Lagrange multipliers and their update scheme, as well as complicating the generation of feasible solutions [2].

[3] A more detailed introduction to Lagrangian Relaxation can be found in [28, Chp 3].

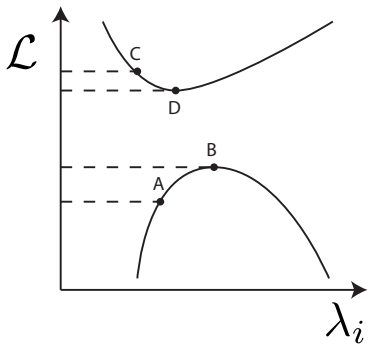


Figure 2.1: Sketch demonstrating the iterative nature of solving Lagrangian Relaxation, recreated from Figure 1 in [29]. The lower curve represents the value of the dual function. Point A would be this value for the values of the Lagrangian Multipliers specified. Point B represents the optimal values of the Langrangian Multipliers, which may or may not be calculable. Point C represents the solution to the original problem with the values of the Lagrangian Multipliers as specified. Point D represents the optimum of the original problem. As the search progresses, point A should approach B. Then C, generated from an increasingly accurate A should approach D, thus converging towards the global optimum.

The bound to optimality and convergence properties of Lagrangian Relaxation led to considerable industry uptake once adequate computing power became available [34, p2]. Many Lagrangian Relaxation implementations [26,30] used the bounds from the dual problem in a Branch and Bound setting. These were outperformed by later Lagrangian Relaxation implementations [32], which did not require the overhead of the Branch and Bound search by ensuring the relaxed solutions are very close to feasible solutions of the original problem.

The Branch and Bound setting was not abandoned however. First introduced in 1960 for discrete programming [35], the idea is to eliminate areas of the search space based on relaxations providing provable bounds on the original problem. For a Mixed Integer Programming (MIP) problem such as Unit Commitment, a suitable relaxation is to let all discrete variables become continuous. It should be intuitive that for a minimisation problem[4] no integer solution can be lower than its relaxed continuous optimum. The relaxed solutions can thus be used as lower bounds on optimality.

When solving for a binary MIP, one branches the complete problem into two sub-problems by assigning a given decision variable 0 in one variant and 1 in the other[5]. These two sub-problems are relaxed for all other decision variables and solved. The relaxation with the lower objective is this then branched on a different decision variable, creating two further sub problems. The minimum of all sub problems' objective functions is branched until either all branches have been expanded, or a found integer solution can be proven to be lower than anything that can occur on other open branches.

For a complex problem such as Unit Commitment this formulation has an immediate benefit. Unlike Lagrangian Relaxation, Branch and Bound does not exploit

[4]Maximisation is possible by noting that no integer solution can be higher than the relaxed solution. Search is altered by branching on the sub-problem with the highest objective.

[5]Branching on continuous variables is also possible. Suppose a relaxation gave $x = 4.75$, one would then branch by adding the constraints $x \leq 4$ and $x \geq 5$. As x is integer this retains all feasible solutions and partitions the solution space into regions which will be bounded in the same way as the binary variables discussed in the main body of text.

problem structure and the relaxation requires no change to the problem's formulation, neither in constraints or objective function. Exploiting problem structure can be of benefit, reducing solution times and enabling an algorithm to tackle problems it would not otherwise be able to solve. However, it can also be a negative, in that it reduces flexibility of the algorithm. There are many variants of Unit Commitment with different sets of constraints for different reserve strategies, piecewise linear vs constant start up times, different cost models, including different types of plants such as hydro and storage, to name a few. Whilst many Lagrangian Relaxation algorithms required considerable modification to handle new constraints [2], continuous relaxation in Branch and Bound requires no modification, allowing for much faster development of advanced formulations of the problem, detailed below.

The downside of a Branch and Bound approach is the time and memory required to solve the relaxations. It requires more memory than Lagrangian Relaxation, whose relaxations typically reduce coupling allowing smaller subproblems to be independently calculated. As computing power continued to increase, relaxing the Unit Commitment MIP to a continuous LP became more feasible. A major reason for the efficacy of this approach is that the simplex algorithm used to solve the continuous relaxation for linear formulations could be warm started, dramatically reducing the solution time of each relaxation [6, Chp 3]. This continuous relaxation, rather than converting to a Lagrangian Relaxation dual, gave a more tractable solution process compensating for the relaxation being harder to solve. Furthermore, with the advancements of multi-core and parallel computing, the inherent parallelism of simultaneously solving subproblems on different branches can be exploited.

Analogous to the difficulty of updating the Lagrange multipliers and selecting the decision variables in Lagrangian Relaxation, Branch and Bound has the difficulty of deciding which variables to branch on. It should be clear that without accurate guidance the search process could enumerate all variables, and for large scale problems, become intractable, as was the downfall of practical Dynamic Programming implementations. In practice, software such as Gurobi and CPLEX have efficient pre-solvers which simplify the problem by taking advantage of problem structure

reducing the chance of intractability.

A major step in preventing the curse of dimensionality afflicting Branch and Bound came from Padberg and Rinaldi [36]. They added extra constraints known as Cutting Planes to a MILP formulation prior to running Branch and Bound to find a solution. The resulting algorithm is termed Branch and Cut, although the two terms are often used interchangeably in modern literature.

An optimal solution to a linear program lies on an extremity of the feasible region. In a MIP, the feasible region of the continuous relaxation encompasses all feasible integer solutions to the problem. The cuts are made to remove non-integer extremities from the feasible region so the relaxation will produce a predominantly integer solution.

The difficulty of this method is in determining what these extra constraints are. The cuts in Padberg and Rinaldi's work were added with specific knowledge of the Traveling Salesman domain being studied, however commercial packages began to develop preprocessing methods which could add similar cuts to generic problems [34, p158].

By adding extra cuts as a pre-processing stage to Branch and Bound, the enumeration of states in the branching phase is greatly reduced. Coupling this with a fast solution method, the warm starting simplex algorithm for linear problems, allows problems such Unit Commitment previously thought of as intractable, to be solved as a MILP without translating into another form. The increased flexibility of Branch and Bound coupled with high levels of accuracy meant it became the standard approach for tackling deterministic short term Unit Commitment in industry.

2.3 Modern Unit Commitment

Today a Mixed Integer Programming (MIP) formulation solved using Branch and Bound is the standard method for Unit Commitment for many system operators. PJM was the first major power system operators in the US to implement such a model for Unit Commitment [37], taking their system live in 2004. It was expected

to save around $60 million dollars annually as a result of the more optimal solutions. In 2009 the California Independent System Operator (CAISO) followed suit and introduced a MIP model for day-ahead Unit Commitment [4], projecting savings of around $52 million. Following a market redesign in 2009, Southern Power Pool (SPP) also introduced a MIP model for day ahead market clearing. ERCOT introduced a MIP model of Unit Commitment after their market redesign in 2009 [38].

As of 2011 PJM, CAISO, SPP, ERCOT, the Midwest Independent Transmission Operator (MISO), the Independent System Opertaor for New England (NE-ISO) all implemented MIP models for some form of Unit Commitment [4, p18,20][6]. The New York Independent System Operator (NYISO) is launching its MIP implementation in 2014.

It would appear that there was slow uptake of MIP models for Unit Commitment, with NYISO still not using one. A plan for NYISO had long been in the works but there are many complications to overcome when implementing a MIP model. As mentioned above, the correct market structure must be in place before day-ahead UC can be used. Market reforms can only come after long periods of negotiations and trials with all parties involved, including SOs, suppliers, regulators and legislators. [5] also mentions the difficulties of migrating existing systems, such as Energy Management Systems, still necessary for operation from old hardware to a more modern setup, typically involving dedicated clusters of PCs. New solvers also have to be integrated with those existing systems.

These American Regional Transmission Operators (RTOs) are a good barometer for the state-of-the-art as they are all not-for-profit organisations with a high level of transparency through FERC. They are also manage some of the largest operator areas in the world [2]. Thus, if technologies perform well on those systems it is likely they perform well, and are in use, in many other smaller systems as well.

MIP modelling for Unit Commitment was not always commonplace. Due to a lack of computing power as other methods such as LR and DP were being used in

[6]Of the other US FERC ISOs, find information about Unit Commitment procedures for Ontario IESO and AEISO was unavailable. Official information with regards to NBSO could also not be found, however work strongly suggests a MIP implementation would be beneficial [39]

industry for Unit Commitment, the prevailing opinion was that a MIP formulation was simply too slow to be of practical use. The 1999 "DIMACS/EPRI Workshop on Next Generation of Unit Commitment Models" was seen as a turning point [2] for MIP models. Bixby, who later went on to create the Gurobi optimisation system, presented work on commercially available systems such as CPLEX and XPRESS-MP which implemented the then recent theoretical work on cutting planes and integer programming techniques to improve performance of MIP algorithms [40].

The conference was followed up by a book of technical papers [34], the first of which summarised the recent improvements in MIP solution methods by assessing the performance of various algorithms on a Unit Commitment problem. They showed an improvement in solution time from 1687 seconds to 98.1 seconds for the then-latest version of CPLEX [34, p6-7].

There was also some small scale industry usage of MIPs beginning to emerge at this time too [34, Chp 9]. A Belgian Utility company created a reduced model, where some of the binary variables are fixed via a sub-problem solved using Lagrange Relaxation, and solved that via a MIP solver. The software XPRESS-MP, used at time of publication by Northern Ireland Electricity, also employed a MIP to optimally solve a reduced version of the problem.

Also of note is that a reasonably competitive Genetic Algorithm (GA) was also presented in that publication [34, Chp 11]. Comparable to integer programming in terms of solution time at time of publication, GAs have seen continued interest within academia. This avenue has not been picked up by industry however, implying the desire for optimality or at least a provable bound to optimality is highly desired by industry.

Since the '99 conference, MIP solution algorithms continued to develop strongly. Streiffert, Philbrick, and Ott presented their work developing the PJM MIP model, the first to be used on a regional scale, in a 2005 paper [2]. They credit the flexibility of the MIP model solution methodology as being an advantage for Unit Commitment. Additional constraints and model parameters can be easily added without any changes required to the solution methodology, whereas LR requires recompilation of the Lagrangian dual, which is a complex problem.

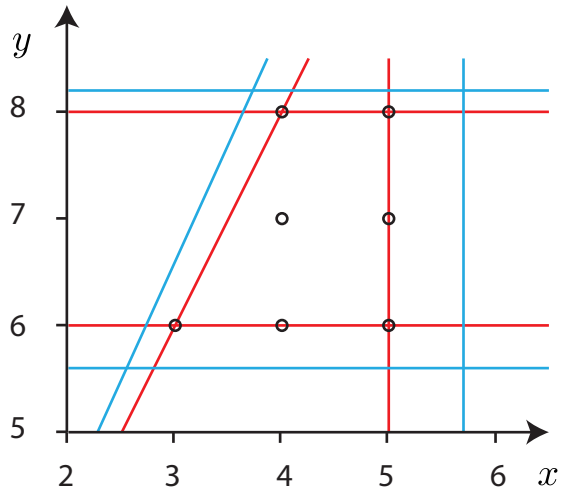
Results were very competitive but also similar to specialised LR methods. The benefit is the MIP can be extended and altered easily without much modification and more complex systems, ("combined cycle units, hydro unit commitment, ancillary services, etc." [2, p8]) can be considered with little modification to the algorithm.

At time of publication CPLEX 9 had been developed and incorporated pre-solve techniques, advanced heuristics and cutting planes. Cutting planes became a great avenue of research resulting in impressive solution time gains for Unit Commitment.

The feasible region for an Integer Programming Problems (IPP) is a discrete set of points in solution space, whereas continuous problems have a continuous feasible region, defined by the inequality constraints. The solution to linear optimisation problems is achieved at an extremity of this solution space where two or more of the planes generated by the inequalities meet. If one can define the set of inequalities for which each vertex is an integer feasible solution, then the IPP can be solved as an LP. Defining this set of inequalities is at best exceptionally computationally intensive and at worst not possible. The continuous region defined by the cutting plane inequalities is known as the 'Convex Hull'. Expressing an optimisation problem as a convex hull drastically reduces the computation time, as a the solution to the continuous problem will also be the integer solution as illustrated in Figure 2.2.

Rajan and Takriti introduced a Unit Commitment model in 2006 [41] with extra inequalities describing the convex hull for the minimum up and down times of generating units. These extra inequalities, despite requiring further binary variables, reduce the solution time by formulating a continuous search space that is in part close to the convex hull. There extra inequalities and binaries do increase the model size (reducing compactness), but the tighter formulation is beneficial and results in a net decrease in solution time.

Recent work by Ostrowski et al [42] presented a set of inequalities which describe the convex hull for the power generation sub-problem of Unit Commitment. This again brought a reduction in solution time. Finally, recent work by Morales-España et al [43] (2013), combines these formulations. Typically a complex MIP model has to make a tradeoff between tightness and compactness. The model introduced in [43] is simultaneously tighter, by introducing all cutting planes from aforementioned



$$0 \leq x \leq 5.7$$
$$5.6 \leq y \leq 8.2$$
$$y \leq 2.2x$$

$$0 \leq x \leq 5$$
$$6 \leq y \leq 8$$
$$y \leq 2x$$

Figure 2.2: Schematic illustrating the convex hull of an integer optimisation problem. Consider maximising $3x^2 - y$ for x and y integer. The problem can be expressed as in either the blue or red box. The region defined by each set of constraints contains the same integer feasible points. Solving a continuous relaxation of the blue problem will give the solution $x = 5.7,\ y = 5.6$. In this trivial example it is easy to see the integer solution is $x = 5, y = 6$, however supposing there were more variables it would be unclear what integer values to set these to, and on what to branch. The solution to the continuous relaxation of the red problem is $x = 5, y = 6$, which is also integer so no further reasoning is required. Because each vertex in the region defined by the constraints in the red box is an integer solution, regardless of the function to maximise or minimise, the solution would also be an integer solution. Thus the red equations define the convex hull of the problem.

papers to close the solution space around the convex hull, and more compact than both, with fewer binary variables and constraints. Test cases on systems from 10s of units to hundreds are presented with the Tight-Compact formulation outperforming (in solution time and gap to optimality) previous formulations from the literature [43, p8] in all but 1 case.

These advances combine to create a hugely competitive formulation for short term thermal Unit Commitment, and it is difficult to imagine another technology which could provide such a level of accuracy in a similar time frame. However, just as the techniques for solving Unit Commitment have evolved the problem itself has increased in complexity. Changes to the grid and power generation portfolio mix have lead to increased uncertainty in the Unit Commitment problem. Quantifying that uncertainty and incorporating it into a successful Unit Commitment formulation is a crucial next step for the power industry. A failure to tackle this problem would result in a system operating either insecurely or at very high cost.

2.4 Handling Increased Variable Generation

Electricity generation accounted for ∼24% of carbon emissions in 2006 and one point projection of electricity consumption expects an increase of 77% from 2006 levels by 2030 [44]. EU governments demand a reduction to 20% below 1990 levels by 2020 as part of the 20-20-20 climate change initiative [45]. This is clearly not possible without a dramatic decarbonisation of the electricity network. Wind generation in the UK alone has grown to 10 GW at time of writing (see [46] for latest statistics) and is projected to increase to 30-45GW by 2030 [47, 48]. The variability of wind and other renewables is in stark contrast to entirely controllable traditional power systems [49]. To reduce the curtailment of wind power SOs must predict wind power output and reduce scheduled thermal output accordingly, adding stochasticity to the UC problem.

Controllability of traditional generation allowed Unit Commitment to be modelled as a deterministic optimisation problem, as discussed above. The aforementioned changes mean traditional formulations of the problem should be reconsidered

to take the variability and non-controllability of renewable resources into account.[7]

One commonly proposed method in the literature for dealing with this is to combine Stochastic Unit Commitment and Rolling Planning.

The principle behind Stochastic Unit Commitment is to discretise the potential outcomes of a stochastic process at a future point in time, e.g. the wind output in 1-3 hours. Each discrete scenario is weighted by a probability and given a cost value based on a Unit Commitment policy. These costs are summed to find the expected cost for that policy. The policy which minimises this expected cost is the solution. The first demonstration of Stochastic Unit Commitment on the approximate scale that would be required in practical applications was seen in 1996 [50], although industry is yet to implement the method due to technological limitations, discussed below.

Rolling Planning is where Unit Commitment is recalculated at regular intervals throughout the horizon utilising updated forecasts for the time remaining. Combining Stochastic Unit Commitment and Rolling Planning for Unit Commitment is very effective as forecasts of the most variable stochastic process, the wind, improve dramatically over a 24 hour period, and recalculating Unit Commitment takes advantage of this improved accuracy.

In 2006 Barth, Brand, Meibom and Weber [51] proposed one of the first models to be used on a realistic large scale system incorporating many market constraints of real world implementations. The model was based on hourly descriptions of demand, generation and transmission, incorporating 4 electricity markets (day ahead electricity, day ahead automatic reserve, 1 hour ahead intra-day electricity, and intra-day non-spinning reserve). Included in this model are variables for wind power producer bids[8], multiple wind power forecasts and actual wind outputs. These variables are the stochastic quantities taking values from the scenario tree to complete the sum of expected costs, and the process is repeated every three hours to take advantage of the expected improvements in forecast.

[7]Note that traditional UC formulations could remain unchanged if larger reserve requirements were imposed, however this would increase cost dramatically.

[8]In reality the prices of generation are not fixed a priori, as is the case when Unit Commitment

Large cost reductions are seen in [51, 55] (2006, 2007) using this method on a simulated multi-region area modelling Scandinavia, Denmark and Germany. The same system, coined WILMAR, is used to analyse the Irish system in [56, 57] (2009, 2011) where savings of around 0.25% over the deterministic solution are shown.

One reason to highlight these studies is the inclusion of real wind forecasts for the horizon considered in [51, 55] compared to generated forecasts and scenarios in [56, 57]. A model is more likely to demonstrate larger savings if it is predicated on a distribution that the data used is generated from. In reality the wind forecast errors may not follow this distribution and both savings and system security will be reduced. Thus, if forecasts are generated rather than taken from historical data, then the savings presented could be exaggerated. It is important ensure these are representative of real world applications including High Impact Low Probability events.

Another difficulty when assessing the efficacy of Stochastic Unit Commitment is the deterministic model used for comparisons. In the work on WILMAR, the authors also give a full description of the model formulation used, in some cases (see for example [58] the unit commitment formulation is not given. If the model used is unrealistic or the baseline model is comparatively less well developed than others in its field, proposed cost savings may be exaggerated.

One downside of these methods which could prevent the realisation of the claimed savings is that the discretisation can not capture the extreme events from the tail of distributions that cause critical problems which should be hedged against. For example in [51, 55] the first 3 hour period is assumed deterministic, the next 3 hour period has 5 outcomes and the final period (can be 3-21 hours depending on the stage of rolling planning) has 10 outcomes, just two further deviations from each previous scenario. Hedging against deviations from these 10 outcomes is done by setting

is presented in much of the literature. Each generation company puts a bid forward for the cost of their generation and, depending on the market structure, a fixed price for generation is found. The stochastic nature of wind power means that bidding for wind power produces can be more complex than for thermal power producers. See [52–54] for examples of Wind Power Producer optimal bidding policy.

reserve requirements based on the wind forecast error being in the 90th percentile. Clearly the true characteristics of a distribution cannot be captured with such a sparse discretisation.

This highlights the ever-present trade off within Stochastic Unit Commitment; too many scenarios and the complexity of the problem increases to intractable levels, too few and High Impact Low Probability (HILP) events aren't captured. This can cause critical problems. 2008 saw ERCOT call for a Emergency Electric Curtailment Plan because of such an event [59]. Unexpected losses of conventional generation and higher than average customer demand, both due to severe weather, combined with a short duration high magnitude drop in wind generation. The event drew much attention because of the role wind power played in it. A failure to compensate for these HILP events would lead to more cut offs, to more customers, more frequently.[9]

With governments in many countries including the US and UK committed to 20% [60] and 25% [46] wind power and renewables penetrations respectively in the grid there is a pressing need to develop Stochastic Unit Commitment methods that are not just theoretically promising but tailored for real world SOs based on current market structures with current equipment levels, to be in use within the decade.

The most recent two FERC technical conferences, where academics and industry expects gather to present and discuss the latest challenges with regards to optimisation in power systems, specifically sought out contributions in the field of stochastic modelling for Unit Commitment. Much of the work discussed below is still in progress and is based on presentations made publicly available online [16, 17].

Of particular interest are the industrial presentations, the results of which are less frequently published than academic papers in journals. A large collaboration project between Sandia National Laboratories, ISO-NE and multiple universities is underway. The key goal is extending the state-of-the-art in Stochastic Unit Commitment from tens of units to low hundreds of units, tackling SO scale problems using real-world data in practical run times [61–63].

[9]These HILP events could be compensated for with large reserves online, this would however be very costly, contrary to the goal of using Stochastic Unit Commitment to minimise cost whilst minimising curtailment of renewables.

One key theoretical goal is the ability to specify a rigorous confidence interval for solution optimality, i.e. what is the probability that the solution given is suboptimal by more than $x\%$. The team presented the problem and outlined their goal at the 2012 FERC conference [61] which was followed up in 2013 by two further presentations [62, 63].

The first of those presentations outlined theory of how to define a rigorous confidence interval in a Stochastic Unit Commitment problem. As the problem is now intrinsically linked to the multiple scenarios it is important to ask how confident one is in the solution presented. The team use historical data of the forecasted load and error in that load interpolated with regression splines to generate an hour load data pattern and error distribution for each hour. There are no assumptions as to the distribution of this error as it comes from the epi-spline interpolation. This avoids the pitfall discussed above where using generated data can exaggerate savings.

The practical difficulty is of course choosing the historical data to sample from, an unrepresentative set for the upcoming weather forecast will not give good results. How to sample from the data such that the scenarios in the sample to be interpolated are a representative collection of scenarios (i.e. chosen to based on what is forecasted for the horizon of interest, not randomly chosen), and what the minimum sample size necessary to give accurate results is, are both open questions.

The key driver of that theoretic work is to provide representative and realistic inputs to the Stochastic Unit Commitment problem. The second presentation at FERC 2013 demonstrated successful implementations utilising initial results from the theory. The WWEC-240 test system used contained 240 bus bars and 140 generators representing a real-scale SO problem. The load forecast model was taken from historical ISO-NE data from 2011 and the wind data came from the EWITS wind data study, both of which were scaled appropriately with a wind penetration of 10%.

Given the scale of the project they have been able to run tests on clusters of computers, similar to the workstations SOs would have available to use. The test systems presented were limited and designed to be illustrative rather than representative. For the no-wind test, with 50 scenarios for load, the stochastic solution gave

approximately a 1.5% saving over the deterministic solution. The 10%-wind case only had 10 load and 10 wind scenarios, which were acknowledged to be too few. No cost results were given but it was noted that solution time on the cluster only increased by around 12%, remaining solvable in practical situations.

It is important to remember these are interim results and indicate that there is great potential for Stochastic Unit Commitment, with practical solution times of around 10 minutes and tangible cost benefits on real-scale problems. Of course this work is not alone. IBM, developers of the ILOG CPLEX system, are looking to tackle problem variants around 1500 units in scale [64] and Princeton University is working on a Stochastic Unit Commitment model for PJM [65].

With work on practical systems ongoing in collaboration with industry, there has been much academic interest in developing systems and models which overcome some of the theoretical restrictions of initial Stochastic Unit Commitment formulations.

As mentioned above there is an inherent tradeoff between numbers of scenarios in a tree and the quality of the solutions presented. Recent work by Sturt and Strbac [66] (2012) develops a Stochastic Unit Commitment model based on user specified quantiles of error. Depending on the user's desires the model could be extremely risk averse, including quantiles 0.95, 0.99 and 0.999 in the scenario generation, or more focussed on utilising all the available renewable generation by including lower quantiles such as 0.01, 0.05, 0.1. It is clearly down to SO's discretion as to how risk averse or 'green' the agenda is for that commitment schedule but this flexibility caters for many situations and would be beneficial.

The system is also designed for year long Unit Commitment simulations and there are many simplifications with this in mind. Of note is the empirical suggestion that later decision are of very little importance in a rolling commitment setting. There appeared to be little loss in quality when system state was modelled constant for extended periods in the latter stages of the planning horizon.

Supposing this system is accurate enough and that enough High Impact Low Probability events are covered by the high quantiles specified, the authors suggest that this system reduces the amount of spinning reserve required on the system. This claim is however predicated on the error distributions being accurate. The author

is unaware of definitive wind forecast models which a consensus of the academic and industrial communities deem correct. Predicating system security on weather models which could well be fallible in High Impact Low Probability events, the very events reserves are put in place to cope with, is of itself a risky strategy.

There is much theoretical work presenting frameworks which remove the need for scenario trees (see for example [67, 68]) however many of these still rely on forecast error distributions and other probabilistic quantities which again do not have a consensually approved practical realisation.

To overcome the practical difficulties of implementing some of the scenario and error forecast distribution models, work by Yu et al in collaboration with ISO-NE is based on a different principle [69]. Research suggests (see for example [70]) that on the 24 hour timescale, typical of Day-Ahead Unit Commitment, hourly wind output can be modelled as a Markov Chain. The transition probabilities for this Markov Chain are precomputed and a collection of states is precomputed

In traditional Stochastic Unit Commitment on specifies a series of scenarios, each of which specifies a value for net demand for each time period and is treated as a single (weighted) deterministic problem. Here the authors encode the wind as being in n states from much lower than expected up to much higher than expected, and transition probabilities are calculated based on historical data. A new MIP incorporating these transitions into both the constraints and objective function is solved using Branch and Bound.

The authors compare their approach to deterministic scenarios and a traditional stochastic model with multiple scenarios. A selection of the results for a 309 unit system are shown in Table 2.1, taken from the ISO-NE network with historical wind data scaled to 20% penetration. They demonstrate that removing the need for a vast scenario tree can result in a drastic reduction in the number of penalised scenarios. The authors also show an ability to handle High Impact Low Probability events such as the 2008 ERCOT incident discussed above. With scenarios and probabilities generating High Impact Low Probability events, their Markovian Model incurs 68% fewer penalty scenarios with only a small increase in cost over the traditional SUC paradigm.

	Markovian	Scenarios	Deterministic
Solution Time (s)	117	361	4
Cost ($)	10,857	10,504	13,206
Penalty Scenarios	80	253	250

Table 2.1: Results for various Stochastic Unit Commitment methods, demonstrating how techniques which are not restricted by scenarios tress can be competitive in solution time and system security.

These were only small test system but could be computed efficiently, reaching a 0.2% gap to optimality on a modern laptop. Supposing this model can be scaled up and network constraints effectively incorporate this may provide a faster and more secure alternative to tradition SUC paradigms.

Finally, work by Alvarado presented at FERC 2013 [71] proposes the idea that the traditional way of considering Stochastic Unit Commitment is not the correct way. All work discussed above minimises an expected cost given some distribution or statistical representation of many scenarios that may happen, resulting in an optimal commitment. Alvarado et al suggest that this is an inflexible and impractical way of describing the problem.

They propose that an optimal strategy, not single commitment schedule, is what should be sought. A trivial example was presented demonstrating how splitting into scenarios encodes the explicit assumption that the future is certain throughout that scenario. This results in sub-optimal solutions for anything other than an exact realisation of a given scenario.

The Markovian model above also manages to negate this assumption by encoding all possible transition values from a given state into its constraints, but any transitions not in the model again are missed. The difference in the strategy case is that rather than the solution being a fixed schedule, the solution is a *conditional* commitment schedule. For each time period, an action based on the probability distribution of a given possibility is stated eg. supposing high wind is realised variables take one set of values, supposing low wind they are another set. A modified LR solution method is proposed and a MIP is acknowledged as a possibility for future

research.

Conclusions

Much of the theoretical work discussed is clearly at an early stage and eventual competitiveness is unknown, however the ideas presented highlight that there are clearly many issues to overcome in developing Stochastic Unit Commitment to securely and efficiently incorporate high penetrations of variable generation. Markovian models demonstrate a promising possibility of overcoming the limitations of scenario tress and a strong ability to handle High Impact Low Probability events without vast increases in solution time. There will inevitably be difficulties in developing these methods, perhaps insurmountable, however the above ideas represent alternative avenues of research which could prove fruitful.

Whilst this work highlights the many open problems that remain, existing commitments to green energy require an immediate step forward beyond the highly efficient deterministic work presented in Section 2.3. Work such as the Sandia National Laboratories collaboration demonstrate quantifiable economic improvements over existing methods on real-world large-scale systems. They also concede there is much work left to do to run the systems with enough scenarios to encompass enough High Impact Low Probability events to ensure security levels remain acceptable.

To conclude, a Branch and Bound MIP implementation of Stochastic Unit Commitment has much promise but cannot yet operate at the scales currently required. This is a similar situation which preceded the widespread take up of Branch and Bound implementations, where despite academic promise and a guarantee of optimality, the problem sizes required could not be solved within a practical timeframe.

Thus there is again the opportunity for new methods to tackle Unit Commitment in interesting ways. The use of Markovian and other methods which do not use discrete scenarios have demonstrated potential, as have new ways of thinking about the problem, trying to determine strategies rather than a single schedule. As market restructuring may be required to fully incorporate some systems dependent on rolling re-scheduling, it may be some time before Stochastic Unit Commitment is widely used in industry.

Chapter 3

Short Term Thermal Unit Commitment with Meta-Heuristics

3.1 Introduction

Deterministic, short term, thermal Unit Commitment is typically formulated as a Mixed Integer Programming (MIP) problem. Classical methods of solving this include the Priority List (PL) Method [18], Dynamic Programming (DP) [19], Lagrangian Relaxation (LR) [29] and Branch and Bound [2] (see Chapter 2.2 for a discussion of industry practices for Unit Commitment). Priority Lists and Lagrangian Relaxation are fast algorithms, but can be sub optimal. Branch and Bound can converge to an optimal solution given enough time, whilst Dynamic Programming offers good solution quality but very poor scaling. Despite long solution times Branch and Bound is the method most commonly accepted to be the industry standard as discussed in Chapter 2.

These classical algorithms can be performed in a strict ordering, with no random components and are so named Deterministic Algorithms. Typically their performance can be improved by estimating which parts of the state space will be best to explore next. This estimation is known as a Heuristic. In Branch and Bound this could be deciding which variables to branch from. Heuristics can be deterministic or

have random components so may not produce the same solution and route through search space each time they are run.

Meta-Heuristic Algorithms are a class of typically problem independent non-deterministic algorithm used to guide search in mathematical optimisation problems towards near-optimal solutions [72, p3]. Despite being used for many years, a single formal definition does not seem evident in the literature, however Meta-Heuristics appears to be an umbrella term for any algorithm which combines multiple simple heuristics or search methods in a non-problem-specific manner. [72] selects 4 definitions from a selection of authors and summarises some the key features:

- Typically non-deterministic.
- Provide no guarantee of optimality but can increase search-space coverage.
- Can include 'learning' features to improve accuracy throughout search.
- Are typically used in situations where computational resources are limited.

Examples of such algorithms include Genetic Algorithms, Simulated Annealing and Tabu Search. Due to their non-problem specific nature and lower memory requirements many such algorithms have been applied to Unit Commitment.

There is a vast body of work in the literature covering all aspects of Unit Commitment. Surveys of most approaches presented have also been published (see for example [12, 13]). These give general introductions to the algorithms used, highlighting prominent authors and articles, as well as brief overviews of some strengths and weaknesses.

As discussed in Chapter 2, industrial applications have moved from classical methods such as LR and DP to Branch and Bound approaches. At the time of Branch and Bound emerging as the new industrial standard, Meta-Heuristic methods such as Genetic Algorithms were also being presented [34, Chp 11], but did not attain the same level of popularity.

Since these early implementations there have been many improvements to the methods used, but still no industry take up. Much of the modern literature in this

sub-field assumes Branch and Bound has inherent weaknesses that cannot be overcome; their solution times on realistic scales are intractable. They are then dismissed claiming the inherent optimality cannot be reached in practical timeframes.

As described in detail in Chapter 2 this is certainly not the case, and there are no signs of current Branch and Bound methods being replaced with Meta-Heuristic algorithms. Having established a recurring feature of little industrial take up and interest, it must be asked why this is so. There is thus an opening within this field for a more in-depth review, discussing more than just the work presented.

It is important to assess work not just on general strengths and weaknesses, but on tangible demonstrations of practical applicability. Some keys areas within Unit Commitment are realism of model formulation and scale of the test systems presented. The MIP formulation is inherently complex and the key features responsible for the complexity of the problem should not be excluded.

A unit's change in output between time periods is constrained by its ramp rate. This constraint couples time periods and thus decision variables, greatly increasing the number and complexity of the constraints in the MIP. If new work fails to include these constraints and does not propose methods, or even intent, to tackle these physical constraints in future work, there is a clear barrier to both practical implementation and demonstrations of increased performance over Branch and Bound.

Due to the complexity of Unit Commitment some instances are inherently easier to solve than others, which is more clearly exposed in larger problem instances. The generation portfolio of the system also plays an important role in determining the ease of solution. In order to show a robust and high level of problem coverage[1], an algorithm must demonstrate performance over a wide range of problem instances, preferably adaptations of real-world demand data, which is freely available in most cases.

[1]Large, high quality, problem coverage is essential for Unit Commitment as a high quality solution must be found for every day for the many years the algorithm would theoretically be in use. This will inevitably encompass a huge range of seemingly improbably combinations of factors making the problem instance seem obscure and pathological but still likely to occur. The results

The inherent stochastic nature of many meta-heuristic algorithms, and lack of a bound to optimality to use as a stopping criteria, mean that multiple runs of the solution algorithm can produce varying quality results. To benefit a reader unfamiliar with the field of meta-heuristics, the variation in solution quality for individual problem instances should be made clear and transparently discussed. This would ensure an inherent distrust in methods lacking a bound to optimality is not exacerbated by a solution method demonstrating at best, mixed performance.

As discussed in Chapter 2, the convex hull of some sub-problems within a Unit Commitment formulation are known, and have been since the mid 2000s. These allow many mid-sized systems[2] to be tackled on a single home PC. It is therefore not unreasonable to expect any relatively modern work to compare performance to Branch and Bound, which by this time was clearly emerging as the industry standard.

Due to the volume of work on Unit Commitment it should be evident that many methods with differing strengths and weaknesses exist. Not discussing burgeoning academic methods could also be seen as an oversight. A new method inevitably cannot compete against well established industrially developed algorithms. Despite this there may be reasons, theoretical or empirical, to suggest it could be successful if given similar levels of investment, and should be included in truly rigorous comparisons.

Thus it is important not just to compare with old versions of the same or similar class of algorithm, but also to the wider academic community and current practices of industry. Without those comparisons one cannot assess how beneficial or detrimental the proposed method is. Furthermore it may appear to a reader that the work is not wholly aware of the field it sits in, lessening the impact of the work.

Clear demonstrations of benefits over existing methodologies are of critical importance to the power industry, where new methods undergo rigorous case studies and multi-year trial periods before being implemented [5]. Not including illustrative

of our test system included such an instance which can be seen in Appendix A.1.1.

[2]See Appendix A.1.1 for a description of the test systems developed for this project which included up to 50 unit systems solved in under 30 minutes using a commercially available solver.

comparisons does not facilitate fair assessments of proposed algorithms.

Such oversights are detrimental to researchers, whose progress in a given sub-field may go unnoticed, and have ramifications throughout the community. Researchers in similar fields cannot fairly assess what the state-of-the-art in a given class of algorithm is, and so may make oversights in the comparisons performed when presenting their own systems. This can lead to a lack of transparency in the field, confusion as to the overall state-of-the-art and thus, the progress of the field as a whole. An unclear message cannot bring change to industrial practices. This reduces the impact of the work as well slowing potential developments in the power systems community, where work of potential could be dismissed rather than invested in to develop systems from academic promise to practical implementation.

Below some of the prominent literature in the field of Unit Commitment using Meta-Heuristic algorithms is surveyed. Only short term Unit Commitment (up to 36 hours in advance) minimising cost, not maximising profit in a competitive environment, is considered. Local searches are surveyed in §3.2, §3.3 surveys global searches, with hybrid methods surveyed in §3.4. Finally, §3.5 concludes the critical review discussing weaknesses in this field and how its impact has had less effect on industry than desired.

What becomes clear is that whilst many articles in the field show academic promise individually, a lack of consistency in model formulation, benchmark algorithms and no discussion of current practices, are key oversights potentially at the heart of why this field has not progressed towards practical interest and implementations.

This review is of benefit to those looking for an overview of this field, such as new researchers to both Unit Commitment and Meta-Heuristics, to researchers within the field looking for examples of common pitfalls when presenting new approaches to this problem, and to more experienced researchers looking for an overview of what issues could need addressing to increase the possibility of industry uptake.

3.2 Local Searches

Local searches slightly vary feasible solutions at each iteration, so each new candidate is within a neighbourhood of its parent Because of the small step changes, optimum solutions can be found quickly providing the initial feasible solution is close to the optimum. Generating an initial feasible solution close to the global optimum so a local search can find it is difficult. Also, memoryless local searches will often get stuck in a local minima.

3.2.1 Simulated Annealing

Simulated Annealing is a search algorithm generally employed in discrete search spaces first formulated independently in [73] and [74], presented as a Monte Carlo method of solving traditional optimisation problems based on the apparent random fluctuations and permutations seen in statistical mechanics.

Annealing is the physical process of heating and controlled cooling of a metal. When heated the electrons have different spin states, if cooled in the correct way the resulting configuration of spin states has the minimum energy. In combinatorial optimisation we observe that a feasible solution is analogous to a configuration of the spin states, and the metric cost is analogous to the total energy. A control parameter is introduced to play the role of the cooling process.

The algorithm proceeds from an initial feasible solution by probabilistic generation of a neighbour, or candidate solution, and probabilistic acceptance of that solution until either a desired accuracy is reached or a computation resource (time, memory etc) is exceeded. The method of variation of a candidate solution is typically different for each implementation whilst acceptance is always controlled by the cooling parameter, forming a "Cooling Schedule", which heavily influences the convergence of the algorithm.

Strengths (+) and Weaknesses (-):[3]

[3]The discussions of strengths and weaknesses throughout this document are our own assessments of the papers.

+ Probabilistic acceptance of higher cost solution prevents stalling in local minima.

+ Not memory intensive as has no memory (i.e X_i is forgotten once X_j is accepted)

+ General applicability [75] means much of the literature can be applied to Unit Commitment.

+ Highly flexible. Varying neighbour generation and probabilistic acceptance functions allows for fast local, or more accurate global, search.

- The often mentioned theoretical result of achieving global optimum with probability 1 as the cooling schedule extends [76] has little impact on practical implementations due to the time require to implement the complete schedule.

- Can spend much time generating infeasible states that must be reasoned about.

- Problem specific generation functions and parameter tuning is required for competitive performance.

Zhuang and Galiana [75] presented one of the first large scale applications of Simulated Annealing to Unit Commitment, using test systems of 100 units. They claimed their implementation was significantly faster than Branch and Bound and Dynamic Programming, and whilst slower than Lagrangian Relaxation, it allowed for the specification of more complex constraints.

Implementation was basic. The initial solution was generated using a priority list method, variation from a candidate solution was implemented by changing the output of a unit i at time period t and adjusting as few other units as possible to ensure the constraints[4] are still respected, and the cooling schedule removed a constant amount from the cooling parameter at each iteration. A minimum value for the cooling parameter was used as a stopping parameter.

[4]As in many early Simulated Annealing implementations only a subset of constraints we considered and rigorously adhered too. A feasible solution in [75] was one where generation limits, min up and down times and must run / unavailability constraints are observed.

Results in this initial implementation were mixed with the authors noting that in some cases the iterative improvement method (i.e. Simulated Annealing without the random acceptance of a more expensive solution) required much fewer iterations to converge to a very good solution. As noted above this is highly dependent on the starting state of the algorithm as iterative improvement will not 'climb' out of a local minima. The method gave solutions of higher quality than Lagrangian Relaxation as expected. No comparison to branch and bound or dynamic programming was given, which is to be expected given the lack of available computing power at time of publication.

Complications included generating the set of feasible solutions (solved via having constraints categorised as easy or hard and allowing the algorithm to consider a solution respecting the easy constraints as feasible). Time periods are considered independent (i.e. no ramp rates are considered). This early algorithm showed promised at the time of publication but clearly further advances were needed to make this a competitive solution method.

Other implementations added more complexity such as non-linear constraints and different cooling strategies (see [77] for example) or tweaking the acceptance of modified states by probabilistically accepting lower cost states (see [78] for example), but it was Simopoulos and colleagues who moved the Simulated Annealing algorithms forward. They developed a more advanced algorithm to that in [75] but at the same scale, 100 units, whose main contribution is as one of the first Simulated Annealing implementations to take ramping constraints between time periods into account.

Ramping rate constraints introduce coupling between the time periods. These constraints break the independence of each output variable, with each one now dependent not just of static bounds (minimum / maximum generation) but also on the value of other decision variables. For a MIP this requires increased reasoning about each variable increasing the computation time, whilst for a meta-heuristic algorithm it complicates the state variation / generation sub-problem, as the random variations will not typically be wholly contained by the range of acceptable outputs at the next time period.

It is tackled in [79] by having the Simulated Annealing algorithm tackle only the on / off scheduling and performing a version of Economic Dispatch to impose ramping constraints. [80] takes the method forward by dynamically selecting the exact solution method for the economic dispatch subproblem.

The on / off scheduling is initially developed in [79] and extended later in [80]. Explicit parameter tuning is used to improve performance and three rules of state variation are given. The cooling schedule is also more sophisticated given by $C_{p,k+1} = \alpha C_{p,k}$, where α is less than 1.[5]

The results showed that the adhering to ramping constraints adds a significant premium to the operation cost, highlighting that implementations which don't model all features will give misleading cost comparisons between methods. Results are given for systems of 20-100 units in size, again proving the plausibility of Simulated Annealing as a Unit Commitment method. Costs are shown to be generally competitive against Lagrangian Relaxation and a selection of meta-heuristic algorithms but the gap to an optimal (or near optimal) Branch and Bound is not considered.

Simopoulos *et al.*. extended their work later in the same year by implementing non-deterministic reserve levels [82]. This paper demonstrated the flexibility of Simulated Annealing and its ability to handle complex constraints but no comparisons to other methods were made so an analysis of Simulated Annealing's competitiveness with these extra complexities cannot be made.

Simulated Annealing used on its own has had significant attention but higher quality results can be obtained by using it as part of a hybrid system. This is a direction research of Unit Commitment has taken recently, and is discussed in Section 3.4.

3.2.2 Tabu Search

Developed by Glover between 1986 and 1990 [83–85] Tabu Search is a local search with an embedded memory structure used for combinatorial optimisation. The memory structure consists of short-, intermediate-, and long-term structures. Rather

[5]This is a common cooling schedule with typical values between 0.85 and 1 [81].

than moving forward by varying a single feasible solution some neighbourhood of the current feasible solution is considered and the next candidate solution is chosen as the best in that neighbourhood. Short-term memory structures retain the previous solutions which become taboo states, so the previous solution is not included in the new candidate solution's neighbourhood. Intermediate and long-term structures are used to bias the search towards predetermined high quality solutions are to reset a search should it become stuck in a plateau. These are typically problem specific.

Strengths (+) and Weaknesses (-):

+ Not necessarily a stochastic method as moves towards best solution in each neighbourhood, therefore does not necessarily suffer from the wide variability found in other stochastic methods.

+ Not required to be linear cost functions unlike MIPs whose solution process becomes much more complex for higher order cost functions.

- Parameter tuning for size of neighbourhoods required otherwise requires prohibitive amounts of memory. Sometimes attributes of a solution become taboo rather than a whole solution increasing the effect of the tabu and reducing the memory footprint.

- Only works in discrete problems spaces as real valued search is vanishingly like to re-generate a previous solution so there is little benefit a taboo list. Competitive Unit Commitment algorithms often separate the discrete scheduling from the continuous dispatch so this general weakness is less of a concern.

Mori and Usami were the first to implement a Tabu Search for Unit Commitment in [86] and [87]. Both of these methods were only tested on small systems of 10 units but became among the first attempt at a 'hybrid system', necessitated by the integer only nature of the algorithm.

Another 'pure' Tabu Search implementation can be found in [88], published in 1998. This time moderately sized systems are tested (up to 26 units, 24 periods). One key contribution of this more in depth publication is the discussion of 4 different approaches to creating the Tabu List as well as varying sized lists. This highlights

the need for domain specific alternations to the black-box meta-heuristic algorithms to gain competitive performance. Of note here also is the use of a quadratic programming regime to solve the economic dispatch, which would allow (given modern computing power) the specification of very complex constraints should 'pure' Tabu Search continue to be developed.

In 2001 Mori and Matsuzaki [89] took the domain specialisation further by using a Priority List of generating units to restrict the generated neighbourhood of each candidate solution, thereby increasing the efficiency of the search. The authors report improvements of Genetic Algorithms and Simulated Annealing on an IEEE 54 unit system. This is a notable achievement and further demonstrates the need for domain specialisation. Lagrangian iteration is used for economic dispatch. For fair comparisons between Tabu Search algorithms it should be clear that the same economic dispatch algorithm should be used so comparisons between papers is not entirely unbiased.

These early papers were published with little computing power so a lack of comparisons is understandable. They highlighted that the search method was feasible and unlike many other classes of meta-heuristic algorithm, the complex constraints of unit commitment could be modelled well as they can be tackled by a separate economic dispatch algorithm. Unfortunately further specialising the tabu-search and combined with traditional mathematical programming methods does not seem to be an avenue explored by the community. Instead Tabu Search has been used extensively combined with other meta-heuristic methods, more details can be found in Section 3.4.

3.3 Global Searches

The global methods discussed below all maintain a collection of candidate solutions. This has a memory cost above the local searches but combined with an often larger random component means the search is less likely to stall in a local minima. The larger fluctuations of a global search mean that more of the state space can easily be covered and the starting state has less impact on the final solution. On the other

hand it can also mean that re-runs of the same algorithm with the same starting state can have much larger discrepancies. Care is needed when assessing the solution quality of these methods to ensure the spread of results is clearly documented.

3.3.1 Genetic Algorithms

Genetic Algorithms are a class of Evolutionary Programming Algorithms; algorithms which find solutions to optimisation problems by mimicking natural selection. In general the algorithms proceed from an initial feasible solution through mutation of a single or selection of candidate solutions to generate a new 'population' of candidate solutions. A metric is defined which determines the 'strength' of a solution. A selection of the strongest solutions are kept and a new population is generated through a crossover ('breeding') algorithm which takes characteristics from multiple solutions. Each member of this new population may also mutate individually before the process starts over.

As with other meta-heuristic algorithms, performance is dependent on the mutation process. Population sizes and the size of the pool of 'strong' solutions to 'breed' from also has a big impact on the performance of a Genetic Algorithm.

Strengths (+) and Weaknesses (-):

+ Searches a selection of points in each iteration not a single point due to population and breeding sizes. Each mutated solution impacts how future solutions will be generated so is more than a collection of parallel local searches.

+ Parallelises well allowing practical problems of large size to be solved using PC clusters.

- Assessing the fitness of each member of a population can be very time consuming so in some cases only samples of the population are assessed.

- Initial population generation can be difficult and time consuming in problems such as Unit Commitment where randomly generated solutions will likely be infeasible. Biasing the initial population generation is possible but a uniformly

random collection is still needed for good coverage of solution space and convergence time.

- Constraints are represented by cost violations i.e. hard constraints are not necessarily observed.

- Parameter tuning (for generation sizes, breeding size, breeding algorithm parameters etc) is required to get the best performance and the optimum value of each parameter will likely vary for different problem instances.

- Two sources of random-ness (mutation and cross over) mean that on problems where optimality cannot be reached practically (such as Unit Commitment) the solutions converged upon can vary quite largely in terms of quality.

Genetic Algorithm implementations of Unit Commitment began as early as the mid 1990s (see [90–92] for example). [91] is a very basic implementation typical of first attempts at using meta-heuristics for unit commitment. No heuristic is used for guidance, performance is only tested on a 6 unit system (later tested on a 9 units system in [92]), Economic Dispatch is considered only to ensure supply levels are above demand and no coupling between time periods is considered. Standard crossover and mutation algorithms were used.

In these early papers favourable comparisons to Lagrangian Relaxation are made but the lack of realistic constraints and small system size means these models are not promising on their own. They highlight that using general methods within a meta-heuristic will rarely provide an accurate solution quickly.

Kazarlis *et al.* [93] presented a Genetic Algorithm implementation around the same time (1996) as the above methods with domain specific genetic operators. These operators cross and mutate large segments of the solution rather than individual entires, which will often generate invalid solutions which break the min on / off constraints. Figures 6-8 therein show the dramatic improvement in convergence to the global optimum over a standard implementation. The authors then impose progressively stricter penalties for constraint breaking and Figure 9 therein shows how this further improves performance.

[93] also presents good rigorous testing results. Systems of up to 100 units are compared with Lagrangian Relaxation and Dynamic Programming (the main contenders given available computing power at time of publication). For each system (10, 20, 40, 60, 80, 100 units) 20 runs were completed (as each run can produce a different solution due to the random nature of the mutation). For systems of 60 units or more the Dynamic Programming could not produce a valid solution (reportedly due to the complexity of satisfying the time constraints in a Dynamic Programming Algorithm) and every Genetic Algorithm ran gave a higher quality solution then Lagrangian Relaxation. No comparison is given to Branch and Bound but given the computing power available at the time of publication this is not surprising. Altogether, this is a much more competitive and complete implementation than many in this field.

An Evolutionary Programming Algorithms (essentially a Genetic Algorithm without crossover algorithms) was developed in 1999 in [94] and also tackles systems up to 100 units in size, suggesting that meta-heuristics do indeed have good scaling properties, backing up the results shown by Kazarlis *et al.*. What is apparent here and throughout the literature however is a lack of common test models and algorithms to allow fair assessment of an algorithm's quality. There are a wealth of papers not mentioned here which all claim to solve the Unit Commitment problem with very similar methods, but without a rigorous comparison against other methods explicitly in the presenting paper it is impossible to assess their quality and claims of success.

Damousis *et al.* [95] present a novel formulation of the MIP and solve for 20-100 unit systems using a Genetic Algorithm. Rather than encoding the on / off state for a generating unit as as string of binary variables, they use a string of integers. If a unit is to be off for 13 periods, on for 22, then off for 13 periods it would be encoded as $(-13, 22, -13)$. Ramping rates are considered and a significant improvement in solution time over a binary coded Genetic Algorithm is shown. Only comparisons to Lagrangian Relaxation are made despite publication in 2004 when a Dynamic Programming and Branch and Bound would both be plausible on the smaller test systems.

It is therefore hard to judge the solution quality of this method, but it would seem to be a competitive formulation for meta-heuristic. By avoiding mutations creating strings such as 0, 1, 0, 1 states which clearly violate minimum on and off times will never be generated. Unfortunately few papers use this formulation so a good idea of the competitive-ness of this method is unknown. This points to a lack of common benchmark problem formulations and solution methods so the relative performance of algorithms and MIP formulations can be transparently compared.

Senjyu and colleagues also developed a standalone Genetic Algorithm [96] with domain specific mutation operators shortly after the above methods (2006). Their operators were more sophisticated, for example by ensuring the mutation of solutions where there was only a deficit in power for a short while only turned on units with short minimum on times. This method was later refined in [97] to give increased solution accuracy. Both implementations were tested on systems of up to 100 units but are only tested against another Genetic Algorithm and Evolutionary Programming implementation, so few rigorous comparisons about relative performance can be drawn.

One common theme of the work discussed above is how continued refinement and specialisation to a domain can dramatically improve the performance of a meta-heuristic method. General 'black-box' algorithms, whilst can be applied to Unit Commitment, do not perform well. This suggests that specialised Genetic Algorithms could prove to be a good solution technique for Unit Commitment.

Unfortunately, rather like the work on Simulated Annealing, ramping constraints between periods are rarely considered, with Senjyu claiming that Economic Dispatch without ramping constraints is already the most computationally intensive part of the algorithm. Rudolf and Bayrleithner [98] develop a Genetic Algorithm which uses Lagrangian Relaxation to perform ramp-rate constrained Unit Commitment but it is only tested on an 8 unit system with import and export contracts. Whilst academically interesting their method is only tested on too small a system to be realistic and takes longer to converge than the reported MILP used to verify the quality of the solution, so has little practical impact.

Standalone Genetic Algorithms can therefore be shown to be competitive com-

pared to a Lagrangian Relaxation implementation on a simplified model of the problem, but their practical implementation is limited. They have however been a common part of hybrid meta-heuristic algorithms, see Section 3.4 for details.

3.3.2 Particle Swarm Optimisation

First introduced by Kennedy and Eberhart in 1995 [99], Particle Swarm Optimisation is another optimisation method inspired by nature, this time by the motion of large swarms of animals such as flocks of birds and shoals of fish. The principle is that a member of a swarm will tend to move from is current position in the general direction of the swarm, with a little random fluctuation. This is transferred into an optimisation model by having each solution as a vector which updates at each iteration by mutating towards the global best, and its previous best solution. Unit Commitment is a binary optimisation problem, so unless carefully constructed operators are used it will suffer the same problems as early Genetic Algorithm methods with solutions with randomly dispersed 0s and 1s violating the minimum on and off times.

Strengths (+) and Weaknesses (-):

+ Unlike Genetic Algorithms it can easily handle continuous variables so is a more flexible solution method.

+ Parallelises well providing communication between parallel architectures is efficient.

- Like Genetic Algorithms assessing the fitness of the population can be extremely time consumings due to the large size needed for good results.

- Handles constraints through penalty functions.

- Binary mutation can easily lead to isolated 1s violating minimum on / off times unless specialised mutators are used.

Many implementations of Particle Swarm Optimisation for Unit Commitment have been given in the literature but few solve a realistic size of problem with a

realistic set of constraints. Gaing [100] proposed one of the first implementations, with two systems of 10 and 26 units. No domain specific operators were used and a unit's ramp rates between time periods were not considered. None-the-less solution quality was shown to be generally 5-10% better than the Genetic Algorithm implementation of [93] as the Genetic Algorithm would converge prematurely despite its domain specific operators.

Ting *et al.* [101] produced an implementation shortly after Gaing which used binary Particle Swarm Optimisation for the unit scheduling and real valued Particle Swarm Optimisation for the Economic Dispatch sub problem. Their results were favourable compared to those given in [93] for other meta-heuristic and classical methods.

The authors claim their "results clearly show the HPSO [Hybrid Particle Swam Optimisation] is very competent at solving the UC problem in comparison to other existing methods." However, ramp rates between periods are not considered, solutions times are not discussed, and the test system is only 10 units in size. The authors give no reason to assume successful scaling of the problem and that the temporal coupling of unit loading won't destroy the performance of their algorithm.

By 2006 at the time of this article's publication, many other meta-heuristic and classical methods had proven performance on much more realistic systems, this highlights how some work does not appear to be fully aware of the state-of-the-art in the field it sits in.

Logenthiran and Srinivasan [102] developed a promising Particle Swarm Optimisation implementation and their work details and compares different Particle Swarm Optimisation implementations; Binary, Improved Binary, and Particle Swarm Optimisation with Lagrangian Relaxation. Ramp rates are considered in this implementation and systems up to 100 units are tested, but there is only comparisons between Lagrangian Relaxation and different Particle Swarm Optimisation models.

Pappala *et al.* [103, 104] develop a Particle Swarm Optimisation algorithm with the same novel integer formulation of the MIP used in [95]. Like [95], the authors had fixed number of integers in [103] but this was restrictive so variable particle size Particle Swarm Optimisation is implemented in [104]. This has the benefit of

reduced variables (reducing the integer section of the particle by up to 88%) and prevents the mutation and cross over operators creating a schedule with isolated 1s. Ramp rates are considered and all constraints are imposed using penalty functions.

In the latter paper the authors run tests on systems ranging from 20-100 units, which are compared with other published meta-heuristic algorithms. The Variable Dimension Particle Swarm Optimisation compared favourably producing cheaper schedules than the 4 comparison algorithms, all also Meta-Heuristic algorithms, on 3 out of 5 occasions and 2nd out of 5 in the other 2. This work seems very promising and it would be interesting to know how close the meta-heuristic solutions were to a branch and bound solution, as well as comparative time to solve between each meta-heuristic and branch and bound.

Both the work by Pappala *et al.* and Logenthiran and Srinivasan demonstrate that Particle Swarm Optimisation may be a suitable for practical applications of Unit Commitment. Both were published in 2010 so are modern implementations and a lack of comparisons to Branch and Bound lessen their impact. It would be interesting to test these algorithms against Branch and Bound and Genetic Algorithms as the impression given is that their Particle Swarm Optimisation implementations produce higher quality solutions than Genetic Algorithms but can take considerably longer. The trade off between solution time and solution quality compared to successful Genetic Algorithms and Branch and Bound over a range of demand profiles would be an interesting study.

3.4 Hybrid Algorithms

It is intuitive that combining a local and global search can remove some of the weaknesses of either method. Typically a local search is embedded within a part of the global search to make a random process biased towards producing higher quality solutions. Whilst these hybrid methods form a large part of the literature the applicability of most of the methods has not yet been rigorously shown, with small test systems and unrealistic models being the main culprit.

Some of the earliest hybrid methods were developed to improve the performance

of dynamic programming. As early as 1989 an expert system (where certain decisions are made to mimic the actions of an experienced schedule form a heuristic) was incorporated within a dynamic programming algorithm [105]. In this early stage of meta-heuristic unit commitment this hybrid provided the rigour of dynamic programming with a reduced computing requirement by providing pre- and post-processing to the Dynamic Programming algorithm.

In [106] an artificial neural network acts as a preprocessor to the dynamic programming algorithm by creating an initial schedule which reduces the search space of the Dynamic Program to only search those states which the neural network has deemed non-certain.

As many of the above algorithms showed early promise, the community quickly began developing hybrid methods which combined algorithms of contrasting properties to overcome conceptual weaknesses of each method.

In 1997, Huang and Huang published a Unit Commitment algorithm combining a Genetic Algorithm with Dynamic Programming and a neural network to generate the initial feasible solution [107]. Tests were run on a model of the Taiwanese Power System with 43 generating units and showed only a slight improvement over the uncombined methods. None-the-less, using Genetic Algorithms to enhanced a meta-heuristic algorithm became very popular.

Mantawy *et al.* [108] published a more successful hybrid algorithm. A Genetic Algorithm was enhanced with Simulated Annealing, by using the Simulated Annealing acceptance test to determine the next generation, and Tabu Search to generate the remaining new population. A 10 unit and 26 unit system was tested and was found to produce slightly cheaper solutions compared to each algorithm on its own and the combination required much fewer iterations to converge (10-25%) than a standalone Genetic Algorithm.

Following in a similar vein to Huang and Huang, Cheng and colleagues developed a Genetic Algorithm with Lagrangian Relaxation [109] which scales up to 100 units but again ramp rates are not considered. The solution times for this algorithm are given however and appear to be approximately linear which is a positive indication that hybrid meta-heuristics could be a practical solution methodology.

Later hybrid algorithms were developed which could successfully handle ramping constraints. In 2002 another Hybrid Genetic Algorithm Lagrangian Relaxation algorithm was developed [110] which does handle unit ramp rates between time periods and runs tests based on the IEEE RTS-96 test system, featuring 96 units [111], without network considerations . A hybrid Simulated Annealing and Genetic Algorithm in 2004 [112] was shown to produce higher quality solutions than classical Lagrangian Relaxation, Dynamic Programming and other meta-heuristics with (in some cases greatly) reduced solution time on systems up to 40 units in size.

As with [101] some of the hybrid algorithms make claims of competitiveness unaware of surrounding work. In 2005 [113] present a Particle Swarm Optimisation Algorithm combined with Lagrangian Relaxation claiming their results on 4-10 unit systems show the "feasibility" of this method. With other meta-heuristic algorithms solving much larger systems at this time a more realistic test model is required to demonstrate that Particle Swarm Optimisation as a field has higher potential than was already demonstrated by Gaing in [100].

Work by Jenkins [114] can be criticised for the same pitfalls as above however it also discusses an important point which is a weakness of many meta-heuristic algorithms. Jenkins presents four hybrid algorithms two of which are praised for their improved accuracy and speed of convergence over the original work by Zhuang and Galiana [75] and by Mantawy [77]. He also explains that these new methods are more robust, providing good performance over a wider range of parameter values than previous work. Parameter tuning can be time consuming and problem dependent so a hybrid algorithm which requires less tuning or can perform well over a large set of problem instances with little alteration to parameter values clearly brings an worthwhile contribution to the field.

Similarly to the work in [88], Bavafa developed an algorithm which combines meta-heuristic search for unit scheduling with mathematical programming for economic dispatch [115]. This work, published in 2009, is a good example of how meta-heuristics have progressed. The IEEE 26 unit 24 bus test system is used, which is relatively small considering sizes used by other authors by this time, but ramp rate constraints, real power constraints and dynamic spinning reserve are all

considered. As with [88], this highlights how combining meta-heuristics with mathematical programming can produce some of the strongest hybrid systems able to solve semi-realistic problems.

3.5 Discussion

Many different meta-heuristic algorithms have been used to solve Unit Commitment in the literature. Despite being able to solve a version of Unit Commitment with little modification to the standard methodology, many of the algorithms published in this area are very uncompetitive without using domain specific operators.

Much of the initial work into Unit Commitment with Meta-Heuristics was done at a time when computing power was both expensive and limited. The quality of classical methods such as Lagrangian Relaxation left a relatively large gap to optimality and the need to find more accurate algorithms with little extra computing power was clear. The key work detailed above filled this gap and proved competitive at time of publication. For more modern work however, better MIP formulations and increased computing power are readily available. Branch and Bound is rarely used as a comparison and modern papers should remedy this as it is commonly accepted to be the industry standard.

It is generally claimed that meta-heuristic algorithms provide faster solutions than deterministic and classical algorithms, although in many cases scalability has not been properly studied and comparisons to Branch and Bound solutions within a known bound of optimality has not been done. Extending an established meta-heuristic algorithm by embedding another has been a popular approach to improve solution quality, combining a search with strong local search properties with one with strong global search properties. This generally improves solution accuracy without adversely impacting solution time but again has not been tested on realistic system sizes, or compared with Branch and Bound for optimality. These oversights combine to weaken the generic claim that meta-heuristic algorithms offer a solution methodology with only small gaps to optimality which is compensated for by very fast solution times.

Branch and Bound can run until a pre-specified maximum bound from optimality is reached. It has been suggested that a gap of around 0.1% is acceptable for day ahead Unit Commitment [2]. If the demonstrated variability of a meta-heuristic implementation is greater than this, then it is highly likely that many of the solutions found by the candidate algorithm will be further from optimality than this. The probability of this occurring cannot be accurately quantified. Thus strong empirical evidence of variability less than this would seem required for industry to begin to consider such a method. Unfortunately this is lacking from much of the research.

3.5.1 Branch and Bound and Industry

In all of the above literature industry standards are rarely discussed. Availability of exact details is limited, probably due to a need for industry to protect the confidentiality of their own class leading methods against competitors. Despite this, common practices can be found, as discussed in Chapter 2.3. However, a collection of models and algorithms representative of industry practices which could be used as a fair means of comparison for novel solution methods would clearly be beneficial to researchers in this area. It would allow researchers to assess how competitive their methodology is, and the wider academic community to assess the state-of-the-art in each sub-field.

Modern PCs have enough power to run Branch and Bound for Unit Commitment Problems of a reasonably realistic size. Furthermore, clusters of workstations can be constructed relatively cheaply, so optimal or near optimal Branch and Bound solutions can be found, and is a common implementation in industry [5, 62]. For meta-heuristic methods to remain relevant modern literature must include more than comparisons with methods such as Lagrangian Relaxation, whose justification for use is negated by the recent rise in readily available computing power.

Highlighting further the oversight of modern papers which do not compare with Branch and Bound is the availability of work such as [2] which describes in full detail a model representative of industry practices. Therein the fundamental model underlying the system used by PJM, which went live in 2004, is discussed but not enumerated. Whilst this does not provide a specific model which meta-heuristics

researchers could copy, it is a clear indication that this is a method demanding serious consideration and comparison to.

Since the emergence of Branch and Bound systems such as the PJM implementation for RTOs, Branch and Bound should have been a factor in most research into Unit Commitment. 2006 saw two further well received papers present efficient formulations of MIPs solved using Branch and Bound [41, 116]. These two papers should have highlighted not only the possibility of modelling Unit Commitment as a MIP and solving with Branch and Bound without the need for super computing and clusters, but also the necessity for comparison to qualify for consideration by industry.

Full details on the advances in MIP and Branch and Cut with regards to Unit Commitment can be found in Chapter 2. Details of the MIP formulation intended for comparison within this project can be found in Appendix A.1.

3.5.2 Benchmark Models and Algorithm

One recurring feature evident throughout this review is that there are no common model formulations. This makes it difficult to draw comparisons between different authors on a level playing field. It would therefore be beneficial to have a collection of models which can be seen as a set of community benchmarks. The issue of benchmarks for Unit Commitment was also discussed in [117] and a method for fair comparisons was presented but has not been taken up by the community.

Like IEEE network models, benchmark MIP model formulations and example algorithms for each class would facilitate transparent comparison between publications without overlooking model discrepancies. Multiple model formulations and algorithms could be used, with different models highlighting specific aspects of the problem. More advanced methods would tackle models with advanced reserve requirements and warm starting units whereas novel and untested algorithms would be compared with early versions of state-of-the-art algorithms on simple models.

The availability of code or precompiled binaries to all in the field would go further, allowing researchers to test each established method against their own on the same workstation, so relative solution times could be accurately discussed alongside

solution quality. In [118] the authors compare their model (on systems of 20-100 units) with Lagrangian Relaxation and an Integer Programming Algorithm. Whilst they can compare costs the authors acknowledge that they can only speculate that the computation time of their algorithm may be considerably longer than either method. A means of verifying this would facilitate clearer assessments of the competitiveness of their method.

The problem is further highlighted by comments such as *"the DP method is capable of finding the global optimal solution, and the method has been utilized as a benchmark method recently"* [101] (2006) and *"Currently, LR method is a widely accepted algorithm in real power system scheduling which is considered as a benchmark for comparing the results."* [102] (2010). Whilst potentially consensual within the meta-heuristic literature, these claims are false at the time of publication. They are also not consistent within the meta-heuristic literature, as no standard Lagrangian Relaxation or Dynamic Programming algorithms are described. Also strong implementations of Lagrangian Relaxation and Dynamic Programming have been known to be out performed by meta-heuristics on large systems as early as 1996 [93], although they have not been developed into industry applications.

Work such as [41–43] provide state-of-the-art implementations with theoretical underpinnings guaranteeing fast convergence and optimality. In lieu of specific industry examples these MIP formulations seem extremely well suited to become benchmark models.

3.5.3 Conclusion

As many constraints are not tackled by most meta-heuristic methods and comparison to industry accepted procedures is severely lacking the impact the field is likely to have on industry is limited despite the academic interest. A collection of benchmark models and algorithms which clearly illustrate the potential of the methods presented in this review will both increase competitiveness and quality of the work within the academic community and drive the field towards more industry uptake and applicability.

Chapter 4

Planning for Optimisation

4.1 Introduction To Planning

Automated Planning, also interchangeably referred to as 'AI Planning' and simply 'Planning', is concerned with sequencing actions to achieve goals[1]. It therefore lends itself well to problems where the actions achieving the goal, the order of the actions, or some associated cost of actions are important.

The archetypal planning example is the "Blocks World" problem. Consider the task of reordering one configuration of distinguishable blocks to another configuration, as in Figure 4.2. The initial state is `A` on `B` on `C` on the table. The goal is to have `C` on `B` on `A` on the table.

To make this a full planning problem all possible facts, known as **Propositions** or **Predicates**, which can be true or false, are listed, as in Table 4.1. The ways in which this system can be manipulated must also be listed. These are known as **Actions**, and each requires specification of **Preconditions**, facts which must be true for an action to be applied, and **Effects**, the results of applying this action.

A **State**, or World State, is a combination of propositions representing a configuration of the world being considered in this planning problem. A planning problem specifies an initial state, which must fully define the system to be manipulated. Of-

[1]The following is a general introduction to the field of AI Planning. For a more complete and rigorous introduction to the subject references [119, 120] are recommended reading.

ten referred to as the Goal State, the goal of a planning problem is a combination of propositions which are to be true for the problem to be considered solved. The goal does not have to fully define a state, as an initial state does. The goal could be just "`C` on `A`", in which case there are many goal states.

An action can be applied to a state if and only if the the preconditions for that action are true in that state. The result of applying an action to a state is to assign each predicate the value given in the effects of an action. Any predicate not listed in an action's effects remains unchanged. To schedule actions to achieve goals is to reason about which actions should be applied to the initial state, in what order, to manipulate the predicates in such a way that the goal propositions are true in the resulting state.

The algorithm which searches for the sequence of actions which transform the initial state into a goal state is known as a planning algorithm, or **Planner**. One can intuitively think of a planner as searching a graph of all possible world states. In the initial state, only a subset of actions will be applicable. Each applicable action can be thought of as an edge, leading to a new node, a new world state. Extrapolating each of these states with their applicable actions creates new edges and nodes, and a full graph of all possible states can be created. Full enumeration of this graph in order to search it would be time consuming and resource intensive, for the 3 block problem this graph would consists of 22 nodes and 42 edges, illustrated in Figure 4.1. Instead multiple approaches to generating and traversing only parts of this graph have been implemented in the literature.

The solution is a path through this graph from the initial state to a goal state. The edges in this graph form a sequence of actions, known as a **Plan**. For the 3 block problem, a suitable plan would be:

```
unstack A from B
put down A
unstack B from C
stack B on A
pick up C
stack C on B
```

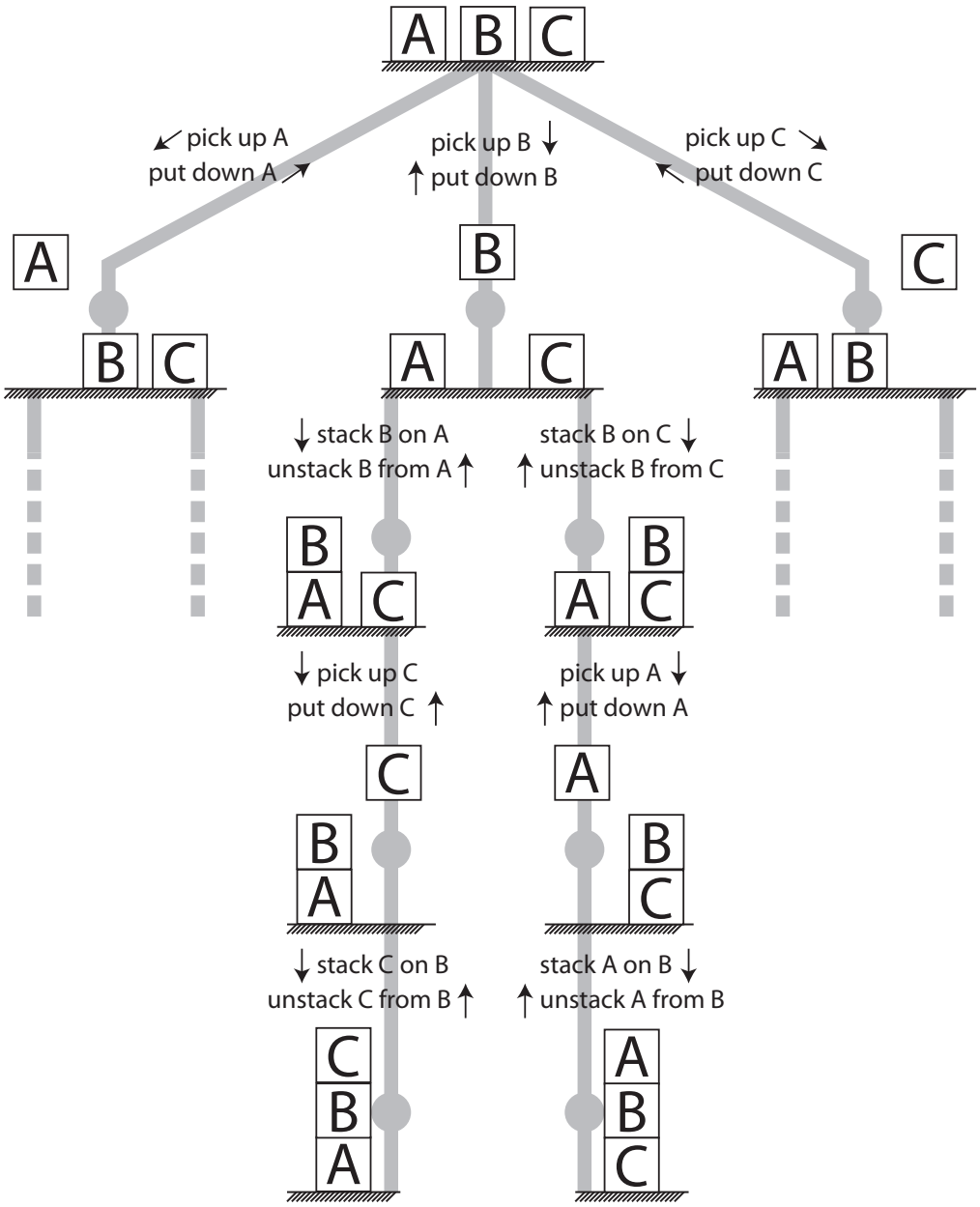
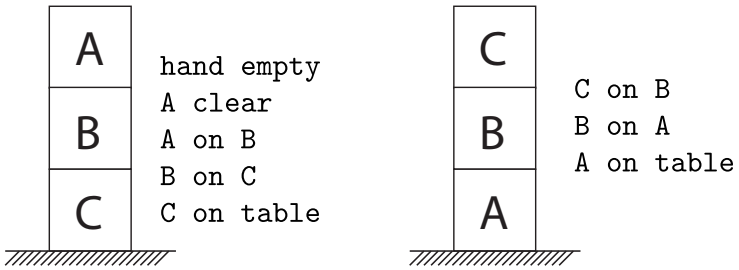


Figure 4.1: A partial enumeration of the search space for a 3 block 'BLOCKS' domain. The four branches not shown will be variants of the two inner branches. Each node represents a state. In this example each action can be undone with a second action, represented by the direction arrows. The complete graph would have 22 states and 42 unique edges i.e. an edge only going in one direction as it represents a single action.



(a) Initial State (b) Goal State

Figure 4.2: Figure illustrating a Blocks World problem with a valid initial state, which describes the state in its entirety, and a goal state which specifies facts which must be true, but does not require full specification of the state.

The main methods for tackling planning problems are introduced through the remainder of this section. Section 4.2 presents a more detailed survey of planning through the development of key planners in the literature. The chapter concludes by assessing the benefit of attempting to tackle Unit Commitment with Automated Planning.

4.1.1 Modelling Planning Problems

Typing and Grounding

The Blocks World problem illustrated above is a very simple example, however it should be clear from Table 4.1 that enumerating all the possible predicates and actions can be quite numerous. In Blocks World there are $O(n^2)$ predicates and actions, for n blocks.

To make modelling planning problems easier, objects **Types** are introduced. Define a `block` type object, then the predicates and actions can be easily listed, as in Table 4.2. Now there are 5 types of predicate and 4 types of actions, regardless of the number of blocks being considered. This allows for a separation between a type of planning problem and the specific problem instance.

A planning **Domain** defines all the object types, predicate types and action types, where each predicate and action can be absolute or take parameters. A

Predicates

A on B
A on C
B on A
B on C
C on A
C on B
hand empty
A in hand
B in hand
C in hand
A on the table
B on the table
C on the table
A clear
B clear
C clear

(a)

Actions

Unstack A from B	
Preconditions:	Effects:
hand empty	¬ hand empty
A clear	¬ A clear
	A in hand
	B clear

Stack A on B	
Preconditions:	Effects:
A in hand	¬ A in hand
B clear	¬ B clear
	A on B
	A clear
	hand empty

(b)

Table 4.1: (a) All possible Propositions for a 3 block Blocks World problem. (b) Example actions from the Blocks World domain. Preconditions are required to be true in a world state for an action to be applicable. The effects are the changes to propositions resulting from applying an action. Other actions are required for each pair of blocks, and actions for picking up blocks from the table and putting blocks down on the table.

Predicates

⟨block⟩ on ⟨block⟩
⟨block⟩ in hand
⟨block⟩ on the table
⟨block⟩ clear
hand empty

(a)

Actions

Unstack ⟨block_A⟩ from ⟨block_B⟩	
Preconditions:	Effects:
hand empty ⟨block_A⟩ clear ⟨block_A⟩ on ⟨block_B⟩	¬ hand empty ¬ ⟨block_A⟩ clear ¬ ⟨block_A⟩ on ⟨block_B⟩ ⟨block_A⟩ in hand ⟨block_B⟩ clear

Stack ⟨block_A⟩ on ⟨block_B⟩	
Preconditions:	Effects:
⟨block_A⟩ in hand ⟨block_B⟩ clear	¬ ⟨block_A⟩ in hand ¬ ⟨block_B⟩ clear ⟨block_A⟩ on ⟨block_B⟩ ⟨block_A⟩ clear hand empty

Pick up ⟨block⟩	
Preconditions:	Effects:
⟨block⟩ on table ⟨block⟩ clear hand empty	¬ ⟨block⟩ on table ¬ ⟨block⟩ clear ¬ hand empty ⟨block⟩ in hand

Put Down ⟨block⟩	
Preconditions:	Effects:
⟨block⟩ in hand	¬ ⟨block⟩ in hand ⟨block⟩ on table ⟨block⟩ clear hand empty

(b)

Table 4.2: (a) lists all predicates in the blocks world domain using typing. (b) Lists all actions in the blocks world domain using typing. Typing is more succinct than enumerating all possibilities and allows separation of the domain and problem instances.

planning **Problem** defines all objects of each type for that instance, and the configuration of the initial and goal states. When a planning problem and goal are combined, all predicates are generated from all types of predicates, and all actions from all types of action. This is known as **Grounding** the predicates and actions. Planners ground the problems before search but this abstraction is very helpful during modelling.

Planning problems are formally described using an Action Language. There are various languages with varying levels of expressive abilities. The simplest planning problems, including that detailed above, stem from research at the Stanford Research Institute in 1971 [121], who proposed the STRIPS problem solver to tackle autonomous navigation. The term later came to encompass any problem of the type defined within that work.

More formally, a STRIPS problem can be defined as [122, p256] a tuple $\langle P, A, I, G \rangle$ where

- P is the set of predicates
- A is the set of actions, each defined to be a tuple $\langle \rho_+, \rho_-, \mathcal{E}_+, \mathcal{E}_- \rangle$ where
  - $\rho_+ \subseteq P$ is the set of predicates which must be true for this action to be applicable
  - $\rho_- \subseteq P$ is the set of predicates which must be false for this action to be applicable
  - $\mathcal{E}_+ \subseteq P$ is the set of predicates which become true after applying this action
  - $\mathcal{E}_- \subseteq P$ is the set of predicates which become false after applying this action
- $I \subseteq P$ defines the initial state. Predicates in I are assumed true, those not in I are assumed false.
- $G = \langle G_+, G_- \rangle$ defines the predicates which must be true (G_+) and those which must be false (G_-) in a world state for it to be a goal state.

In 1987, Pednault propose a more expressive language, ADL. This contained many differences to STRIPS including negative preconditions, object types (discussed below), and actions with conditional effects [123]. Initial states in ADL could now contain predicates with unknown values. Later developing into the larger field of probabilistic planning this advancement has great importance when developing systems for online use. Technical details of probabilistic planning are beyond the scope of this project but an introduction can be found on page 80.

Planning Domain Definition Language, PDDL, was developed in 1998 [124] with the goal of standardising planning languages in preparation for the International Planning Competition (IPC) (see Section 4.1.3 for details). As that competition developed so did the expressive ability of the language, with new features for each competition bringing an update to the language. There have also been many non-competition extensions to the language, typically developed by an individual research team for a specific domain and purpose (see for example [125, 126]). Due to its wide spread use, especially within the IPCs, it is the language used throughout this project.

PDDL Introdution

To provide a general introduction to PDDL, the above Blocks World planning problem is detailing in PDDL here[2]. A PDDL domain file is defined as such by the opening line

```
(define (domain BLOCKS)
```

This opens the definition file and assigns this domain a unique identifier. When a problem file is specified it is assigned a domain name, this identifier indicates problems and domains are intended to be compiled together. For ease of parsing, each entity in PDDL is enclosed in parenthesis. The entire domain definition is enclosed as such, hence the leading '`(`'. The matching closing '`)`' completes the domain definition.

[2]Language specifications for all versions of PDDL can be found in references [124, 127, 128]

The domain definition then continues by listing any objects and their types. In Blocks World there is only one type of object, a block, so its inclusion is optional, but in more complex domains discussed later there can be many object types.

```
(:types block)
```

The `:types` code indicates we are listing the object types used in this domain. Predicates and actions which take parameters are then required to detail what type of object their parameter is.

A hierarchy of types can be defined. The following is taken from the IPC 2002 competition domain 'Depots'[3]

```
(:types place locatable - object
        depot distributor - place
        truck hoist surface - locatable
        pallet crate - surface)
```

The '`-`' indicates there is a hierarchy between the types on the left and the right, the left hand types being subtypes of those on the right. An action or predicate which lists a parameter as `place` can take either a `depot` or `distributor`. Conversely a subtype parameter cannot take its super type. In this domain this allows the specification of a global `Drive` action which can be between any two `place` objects.

After declaring the types used in the domain, the predicates must be listed.

```
(:predicates (on ?x ?y - block)
        (ontable ?x - block)
        (clear ?x - block)
        (handempty)
        (holding ?x - block))
```

The section specifier `:predicate` instructs the reader that the following are predicates. Each predicate is enclosed in parenthesis, parameters are denoted by a question mark. If a parameter is followed by a '`-`', all parameters preceding the type

[3]The IPC 2002 and two competition data is no longer available online but was provided for this project by the competition organisers.

specification up to another type specification are assumed to be of that type. In this example the `on` predicate requires two parameters each of type `block`, whereas `handempty` requires no parameters as it is not object specific.

Finally each action is to be defined.

```
(:action pick-up
        :parameters (?x - block)
        :precondition (and (clear ?x) (ontable ?x) (handempty))
        :effect (and (not (ontable ?x))
                (not (clear ?x))
                (not (handempty))
                (holding ?x)))
```

The `:action` specifier separates this element of the domain definition, enclosed in parenthesis, and the three elements of an action are all subsequently defined. The parameter has a type specifier in the same syntax pattern as a predicate. Preconditions and effects have their own specifiers and each is proceeded by a collection of predicates. When used in this context, the predicate is listed with a parameter taking the same name as one from the list of parameters. No type specifier is required as each parameter should take a unique name in the parameter specification.

It should be clear from the above example and Table 4.2 how the remaining actions of the Blocks Domain are given.

A specific problem instance is conventionally defined in a second file, allowing for multiple problem instances to use one domain file. This makes it easier to automate and allows researchers to run applicable problem instances without requiring specific knowledge of complex domains, perhaps developed by colleagues.

A problem instance file for the Blocks Domain could be as follows:

```
(define (problem BLOCKS-3)
        (:domain BLOCKS)
        (:objects A B C - block)
        (:init (clear A)
                (on A B)
```

```
            (on B C)
            (ontable C)
            (handempty))
    (:goal (and (on C B) (on B A))))
```

It begins with a definition statement similar to the domain file. The `:domain` specifier must link to the domain definition identifier used in the domain file and indicates compatibility. The `:object` specifier enumerates all objects of each type in this instance.

The initial state is given with the `:init` specifier. Here each predicate listed is true. One could write `(not (on A C))` to fully define the initial state. Including all negative predicates would lead to unnecessarily long problem definitions and so any predicate not listed is assumed false.

The goal state is defined using similar syntax to action preconditions and effects. The `:goal` specifier is followed by a union of predicates indicated by '`and`', with each predicate inside parenthesis. This concludes the basic introduction to PDDL. Further details will be added for temporal and numeric features and full PDDL specifications can be found in references [124, 127, 128].

The purpose of defining a common language is not just for ease of communication between researchers, but also to facilitate sharing domains and problems and comparing computer simulations. If all planning problems are expressed using a common language, planning software only needs to be able to parse one language and common libraries can be shared. This means researchers have to spend less time working on parsing code and more time developing planning algorithms and software, speeding up the development of the field. Furthermore it removes the need for researchers to develop their own languages with concerns of expressive ability, efficiency and reusability. It is also key to the International Planning Competition (IPC), which provides a set of benchmark problems for clear comparisons of the state-of-the-art in planning. Further details of the IPC are given in Section 4.1.3.

As mentioned above, to help make parsing easier PDDL, each section is enclosed by matching parenthesis `( ... )`. This includes everything from action predicates to the whole PDDL file. Also, key word specifiers such as `type`, `precondition`,

`action` etc are all prefixed with a colon, `:`. This helps the parsing code by introducing a rigorous specification to the language.

Another helper for the planner is the `:requirements` specifier. This comes after the domain definition line and lists the features of PDDL that the domain uses. In simple STRIPS problems there are few specifiers, `BLOCKS` requires only `:typing`. These requirement specifiers help to separate the development of domains and the development of planners.

As planning developed as a field many new features were added, discussed in more detail below. PDDL and other languages expanded to incorporate these new features. The requirement specifiers allow existing planners to gracefully exit if new constructs cannot be handled when tackling modern domains, rather than crashing with unexpected behaviour.

A further benefit is to facilitate development of feature specific planners. Whilst numeric fluents and durative actions (detailed below) were both introduced in 2002 as part of PDDL 2.1 for the 3rd IPC [127], temporal and numeric planning have very different requirements. Successfully tackling each feature separately required much research before the two could be handled together. Much research is still ongoing and is discussed further in Section 4.2.

Requiring a planner to handle all possible features of the language would have stalled the development of the field. It would also prevent the development of specialised planners. As detailed below, handling both temporal and numeric features of a planning problem requires much advanced reasoning. Many problems will only require the modelling of one or the other, and so more efficient planners can be developed which only hand one or the other.

For this reason whilst many problems can be expressed in PDDL, there is no guarantee that a planner exists which can tackle that domain. This highlights how Automated Planning as a field is still in relative infancy compared to other methods such as Mathematic Programming. It is thus important to have realistic expectations about the immediate applicability and performance of Planning on a domain such as Unit Commitment, which has received much attention from highly developed methodologies over many decades. There are however many reasons to

be optimistic about it as an approach, discussed further in Section 4.3 and Chapter 5.1.

4.1.2 Numeric and Temporal Planning

Whilst many problems can be expressed purely propositionally (variants of logistics planning, planning factory operations and card games were all tackled in the 2nd IPC [129]), more realistic problems typically require numeric features or a timeline. A realistic logistics domain may contain different transport methods which achieve tasks in different time periods with different fuel cost. It is then practical to search for the solution which minimises cost whilst respecting deadlines.

The introduction of temporal and numeric features to PDDL 2.1 marked a significant increase in the scope of problems that could be modelled as Planning problems [127] over existing languages. The community proceeded to develop increasingly complex domains which could be constructed using PDDL and its later variants.

This section explains the concepts behind numeric and temporal planning in more detail.

Numeric Planning

Sometimes referred to as **Planning with Resources**, planning with numeric quantities presents a number of challenges. The first is state representation. Suppose a world state definition is extended from a subset of true predicates to be a tuple: $\langle \mathcal{P}, \mathcal{V} \rangle$ where $\mathcal{V} := \{v_i\}$ represents a collection of numeric variables. Whereas a purely propositional state space can be fully enumerated as all possible subsets of predicates, the variables in $\mathcal{V}$ could take any range of values, integer or continuous. Careful thought for how the search graph is constructed is necessary. The second is the heuristic used to search the graph, which is discussed in further detail below and in Section 4.2.3.

It is important to think about the reasons numerics are being integrated into the planning framework alongside addressing how they are integrated. The first key reason for introducing numerics into the planning framework is to constrain the

problem to a realistic use of resources. There are many real world problems where something can only be done if there is sufficient resource available.

A typical Logistics domain can be found in the 2nd IPC held in 2000 [129]. This has the requirement specifiers `:strips` and `:typing` so is purely propositional. In it, multiple packages are to be delivered to multiple locations by trucks or airplanes. As there are no numerics the planner cannot schedule actions based on distances or fuel costs, quantities which would be important in real world applications.

The 2002 IPC introduced a more realistic logistics domains [130, p46] including a numeric variant. Packages now had weights associated with them, and vehicles had weight limits. This introduced a much more realistic component to the problem, namely assigning the right packages to the right vehicles.

Numerics in PDDL 2.1 are known as **Fluents** but represented with the `:function` specifier. They are listed after the predicates and use the same syntax. The following is taken from the 2002 numeric 'Depot' domain[4]:

```
(:functions
        (load_limit ?t - truck)
        (current_load ?t - truck)
        (weight ?c - crate)
        (fuel-cost))
```

The restrictions on loading the vehicles are then incorporated into loading actions.

```
(:action Load
 :parameters (?x - hoist ?y - crate ?z - truck ?p - place)
 :precondition (and (at ?x ?p) (at ?z ?p) (lifting ?x ?y)
          (<= (+ (current_load ?z) (weight ?y)) (load_limit ?z)))
 :effect (and (not (lifting ?x ?y)) (in ?y ?z) (available ?x)
         (increase (current_load ?z) (weight ?y))))
```

[4]The IPC 2002 and two competition data is no longer available online but was provided for this project by the competition organisers.

The syntax for numeric expressions in PDDL is perhaps not the most natural for reading but is easier to parse in code. An expression is defined as a relation or operator sign followed by two fluents or other expressions. The key words `increase`, `decrease` and `assign` apply the second object to the first in the intuitive manner. Relational operators are used for preconditions and operators or key words are used for effects. A full exposition of the syntax can be found with the original language specification [127, Chp 3-5].

In the above example, the precondition

```
(<= (+ (current_load ?z) (weight ?y)) (load_limit ?z))
```

is equivalent to the algebra

$$l(z) + w(y) \leq L(z)$$

where $l(z), L(z)$ are the current load and maximum capacity of the vehicle z and $w(y)$ is the weight of the package y. Similarly the effect

```
(increase (current_load ?z) (weight ?y))
```

is equivalent to the algebra

$$l'(z) = l(z) + w(y)$$

The final fluent in this domain `(fuel-cost)` hints at the second key reason for introducing numerics into the planning framework; to provide a metric on which to assess plan quality. In the Depots domain a `Drive` action increases `fuel-cost` by a fixed amount, independent of the places driven between and the vehicles driving.

This could be more realistic by extending the IPC Depots domain, adding further fluents `(distance ?x ?y - place)` and `(mile_cost ?v - truck)`. The `Drive` action could then have the following effect

```
(increase (fuel-cost) (* (distance ?x ?y) (mile_cost ?v)))
```

Each drive action now has a more realistic cost function.

To specify that this cost function should be taken into account when constructing the plan a plan metric, on which quality is tested, must be specified for the problem. This is done in PDDL 2.1 using the `:metric` specifier in the problem definition. The

metric can be any numeric PDDL expression but planners may only support a subset of expressible functions, only linear functions for example.

Adding the line

```
(:metric minimize (fuel-cost))
```

to the problem file will mean plans with lower fuel costs will be preferred. Support for metrics, especially complex functions, is a problem in planning, with many heuristics and search algorithms unsuited to this kind of optimisation. Discussed further in Section 4.2.3.

Temporal Planning

The 3rd IPC in 2002 presented many domains which also had natural temporal extensions. It is common for a real world problem to have some interest in the temporal aspects of a plan. A simple factory scheduling may not be concerned with how long processes take as there may only be one process to achieve one goal so minimising construction time may not be a problem. A more realistic variant would be to say that there are multiple products requiring the same machinery and how can processes be scheduled to minimise construction time of a series of different products.

One common temporal domain, used in the IPC and earlier, inspired by space projects is to schedule the actions of various rovers to perform set actions and convey those results back to a base station [130, p49] [131, §12]. Travelling between locations takes time, as does warming up equipment to complete tasks. The problem is made complex by the restriction that communication can only occur at certain times (representing when satellites are aligned and visible). Tasks must the scheduled concurrently to ensure the most information can be gathered for each communication period, i.e by warming correct equipment whilst other tasks are completing to ensure other tasks can begin with minimal gaps.

In PDDL actions which take time are known as **Durative Actions**. In a temporal domain each action gets a timestamp applied to it when added to the plan, and has either a duration specified in the domain or is instantaneous and therefore

as a duration ϵ close to 0, necessary for technical reasons of implementation. The plan is now a sequence of actions each with a timestamp and a duration sitting on a continuous timeline. Planning without explicit reference to a timeline is formally referred to as **Sequential Planning**, and planning allowing concurrent execution of actions at the same time step, but not mapping to a timeline, is referred to as **Parallel Planning**, separating it from this more complete approach to handling time and duration, referred to as **Temporal Planning**.

Durative actions take parameters in the same way as instantaneous actions. The duration can be fixed, parameter dependent, or within a range of time. Note that a range must have an upper bound. Examples include

$$d = 5$$
$$d = 2 + 3 * \langle \text{function} \rangle$$
$$\langle \text{expr} \rangle \leq d \leq \langle \text{expr} \rangle$$

Action preconditions now have 3 qualifiers attached to them. The condition must hold at either the start of the action, throughout the actions duration, or at the action's end. For an action occurring at time t and running for a duration of d, the precondition qualifiers for a predicate p act as follows

$$\begin{aligned} \text{at start} &\implies p \text{ is true at time } t \\ \text{over all} &\implies p \text{ is true though out the open interval } (t, t+d) \\ \text{at end} &\implies p \text{ is true at time } t+d \end{aligned}$$

An action effect, e, has similar qualifiers

$$\begin{aligned} \text{at start} &\implies e \text{ is applied at time } t \\ \text{at end} &\implies e \text{ is applied at time } t+d \end{aligned}$$

The syntax for durative actions is a natural extension of instantaneous actions, expressed in PDDL 2.1 as

```
(:durative-action <action_name>
 :duration (<relation> ?duration <expression>)
```

```
:condition ( and
       (at start (<condition>))
       (over all (<condition>))
       (at end (<condition>)))
:effect ( and
       (at start (<effect>))
       (at end (<effect>))))
```

where `<relation>` is `<=`, `=`, `>=` and a range is expressed using the same '`and`' syntax of effects and preconditions. Multiple conditions and effects can be listed.

Two further constructs which are common in more complex temporal domains are **Timed Initial Literals** and **Timed Initial Fluents**. These specify a predicate or numeric effect which occurs at a specific point in the time line. Expressed in the problem file, the syntax specifies a specific time in the timeline and the effect, typically occurring after the initial state has been declared.

```
(at 10 (<predicate>))
(at 15 (<expression>))
```

where `<predicate>` and `<expression>` take the same syntax as action effects.

Temporal Planning is very expressive in PDDL 2.1 and many problems which are easy to express are quite complex to solve. One particular difficulty is **Required Concurrency**, discussed further on page 106. Formally defined by Cushing *et al.* [132], required concurrency is where one action must be scheduled during the execution of another. This typically arises when one action requires a resource which is only made available throughout the duration of another action.

The following 'Match Domain' (detailed in [133, p13] for example) is a simple example.

```
(define (domain Match)
 (:requirements :durative-actions)
 (:predicates (fuse_broken)
     (fuse_fixed)
     (match_unused)
```

```
    (have_light))
 (:durative-action light_match
    :duration (= ?duration 10)
    :condition (and (at start (match_unused) ))
    :effect ( and (at start (have_light))
        (at start (not (match_unused)))
        (at end (not (have_light)))))
 (:durative-action mend_fuse
    :duration (= ?duration 8)
    :condition (and (fuse_broken)
        (over all (have_light)))
    :effect (and (at start (not (fuse_broken)))
        (at end (fused_fixed)))))

(define (problem simple_match)
 (:domain Match)
 (:init (fuse_broken)
    (match_unused))
 (:goal (and (fuse_fixed))))
```

The problem is trivial and the solution is to schedule `mend_fuse` between the start and end of `light_match`. Despite this many of the techniques initially developed to tackle temporal planning would incorrectly solve this problem, and planners which can treat this rigorously have only recently emerged [132, 134]. This highlights how modelling planning problems and developing algorithms to solve them can both be challenging.

Finally, a domain can combine numeric and temporal features. A more realistic logistics domain may contain different transport methods which achieve tasks in different time periods. It may then be a requirement to minimise cost but respecting temporal deadlines. The modelling requirements of such problems combine the syntax for temporal and numeric planning presented above. The interaction between numeric resources on a continuous timeline however requires complex reasoning [133] and is discussed further in Section 4.2.3.

Advanced Planning

Automated Planning has developed beyond temporal and numeric planning. All the examples above assume a fully observable and deterministic world. More realistic models may include unknowns in the initial state, partially observable world states, non-deterministic effects and ADL-style conditional effects based on state realisation. These topics are beyond the technical scope of this report but are introduced here for completeness.

In **Planning Under Uncertainty** or **Probabilistic Planning** action effects are grouped and one of the groups occurs with a given probability. This allows for problems with probabilistic effects in fully observable worlds to be described. Examples from the IPC in 2004 include logistics domains where vehicles may not end up in the correct places and airport models where passengers may end up on the wrong flights [135].

The motivation for planning under uncertainty was more clearly realised two years later. As noted in the conference literature for the 2006 IPC, the main reason for considering planning under uncertainty is for use as part of an online decision making process [136, p4]. The planner's role would be to take a domain representing some ongoing process, and given the current state and a set of goals, produce a plan, the first few actions of which would be enacted before re-planning based on the realisation of those actions.

Related to planning under uncertainty is **Machine Learning**. This is the idea that any automated system solving a problem with some notion of quality will improve its performance as the number of problems solved increases (see [137, p2] for a more formal definition). Moving to an online non-deterministic setting, an automated system can update aspects of the model and / or data about the world based on observations and realisations of actions. Whilst it is used in everyday software such as email spam filters and handwriting and speech recognition [138, p1-2], it has only recently begun to be incorporated into planning frameworks.

The IPCs in 2008 and 2011 both had learning branches but these were more for initial plausibility testing, data analysis and identification of any key directions to focus future research [139]. Domains included probabilistic variants of archetypal

planning problems such as blocks world, logistics and space applications.

Modern applications include work where planners would interact with people [140, 141]. Inherently non-deterministic but unable to be modelled by probabilistic functions this work presents its own challenges. One example from the latest International Conference on Automated Planning and Scheduling (ICAPS) is a planning system used as a bartender interacting with customers [141]. The tasks the planner should perform are relatively simple but inferring what those tasks are from human interactions requires more complex reasoning and modelling.

Other work demonstrates a system designed to help vulnerable people complete potentially dangerous day to day tasks [140]. Ironing, cooking and running baths can all be potentially dangerous should a person, such as a dementia sufferer, leave them unfinished. The planning system is required to schedule the tasks but not interfere with the patient, instead providing gentle reminders at critical points allowing the patient to feel in control. The system has to learn and respond to unexpected human actions and individual patient traits to ensure they remain safe.

Both Machine Learning and Planning Under Uncertainty are important if Automated Planning is to be used as part of an online system, which would be the case were planning to be proposed as a suitable method in a Rolling Unit Commitment setting.

Searching for a Plan

The modelling of planning problems has been discussed in detail, however the search process has received less attention. This is because there are many different approaches, and few established best practices. Many algorithms, especially tackling complex domains, display good performance on only a small number of domains. There are however some regularly used approaches introduced here, but many are specific to a given planner and details are left to the more indepth literature review in Sections 4.2.2 and 4.2.3.

Planners can have different search criteria. Searching for *any* feasible plan, is known as **Satisficing Planning** [142, p552]. A satisficing planner presented with a `:metric` in the planning problem may use this metric to guide search but does not

provide a guarantee of producing the best plan with respect to that metric.

An **Optimal Planner** is a planner which guarantees to find the plan with the best possible value of that metric. Optimal planning has received much attention, as detailed in Section 4.2.2, but has made little progress in temporal or numeric planning. As such optimal planners that have been developed typically minimise the number of actions in a plan, known as minimising the **make-span** of the plan.

There are multiple methods for searching for a plan. **State Space Planning** searches the graph of world states as described above. Common algorithms include Enforced Hill Climbing and A* search (see Chapter 6.3 for a detailed introduction to the algorithm used in the POPF planner). As important as the algorithm used to search the graph is the heuristic used.

Most planners make use of a **Relaxed Planning Graph** (RPG) heuristic. This is a simplified version of the full world state space graph created from the current state in the search and expanded under simplifying assumptions until a goal state is reached in this simplified space. Properties of this simplified graph are used as guidance to inform the search of the true graph. Example simplifying assumptions and heuristics employed in the literature are discussed in Section 4.2.3.

Partial Order Planning allows actions already added to the plan to be promoted (or demoted) to earlier (or later) positions in the plan throughout the search process. Search can start from an empty or pre-populated (but inconsistent) plan, and advances by making alterations at each step to fix any inconsistencies, such as missing goals, unsatisfied action pre-conditions or mutex relations[5] [143, 144].

There have been many other approaches to planning including Constraint Satisfaction [145,146], Model Checking [147,148] and translating to other existing solution methodologies including Satisfiability [149] and Integer Programming [150].

[5]A mutex relation is a group of contradictory facts that cannot be simultaneously true. In planning a predicate cannot be both true and false. In the preprocessing phase of many planners mutex relations are analysed by considering action effects and what groups of predicates can never be simultaneously true or false.

4.1.3 The International Planning Competition

The International Planning Competition (IPC) was set up alongside the International Conference on Artificial Intelligence Planning Systems (AIPS) in June 1998. Its goal was initially to help unify planning research through the introduction of a common language, PDDL, help identify research goals to widen the scope of the field by identifying and collating challenging benchmark problems, and advertise the progress of the field to the wider community [151]. The competition has since matured to provide a forum for rigorous comparison between the state of the art in Automated Planning on challenging benchmark problems [130].

Since the first competition it has become a biennial event driving forward the development of Automated Planning and empirical methodologies for tackling increasingly complex real-world problems. The following discussion of the IPC is based on summary publications analysing the results of the competitions [130,142,151–153] and official competition websites [154].

The competition procedure was to present a series of planning domains created by the competition organisers to the planning community. Planners would then be submitted and the organisers independently run those systems on the same workstation. This allowed the development of domain specific planners, whose algorithms were coded with knowledge of the problem. This was removed in the 4th competition in 2004 to focus on general applicability rather than specific systems, which was deemed more useful to the wider community.

The planning system for Unit Commitment presented in Chapter 6 is a domain specific system, with a specific heuristic making use of a inherent properties of the domain. It is hoped that the ideas presented will motivate further research into similar domains and that a more general domain-independent approach could be developed, as has been shown throughout the development of recurring entrants in consecutive IPCs [142].

As well as providing an exposition of the state of the art at time of occurrence, the competition summaries noted above enable a chronological study of the development of planners. The competition initially focussed on sequential planning [151,152] both satisficing and optimal. IPC 3 in 2002 introduced numeric and temporal planning

and began the focus on real world applications [130]. Separating satisficing and optimal planners marked the diversification of the field.

There were two types of planners entered in IPC 3, 'automatic' planners which required no extra input for each domain, and 'hand crafted' planners which required some developer input for each individual domain. Here we only discuss the automatic planners as hand-crafted planners fell out of favour due to the extra work required to maintain, update, and tailor to each domain, and were omitted from later competitions [142, p523].

Increasing domain breadth continued with later competitions. Since the 4th IPC in 2004 there have been multiple tracks in the competition alongside the deterministic planning considered in this project [154]. The 4th competition had a probabilistic planning track [142, p520] and introduced Probabilistic-PDDL (PPDDL).

Alongside widening the breadth of domains, IPC 4 also specified a focus on more realistic domains stating that a key feature of these benchmark problems should be to be oriented at applications [155]. This can be seen in that the key oversight of the 2002 Satellite Domain[6], which was stated to be not considering that the antenna on earth are only visible to the satellites, and therefore can only communicate with earth, during certain time windows, was incorporated into the domain and included in the 2004 competition.

Later competitions began adding other language features rather than pushing further towards tackling real-size optimisation problems. The 5th competition in 2006 [153] introduced PDDL 3.0 allowing for specification of soft goals and preferences. Plan quality may not simply be finding the best metric value of a plan. A logistics planner may prefer solutions using one type of vehicle because it is easier to repair or a route because it is less likely to be congested. These do not have numeric consequences that could be easily encoded into a planning domain and solutions which do not respect these desires are still valid plans. Encoding these as soft constraints or preferences allows the user to guide the planner in domain specific ways without requiring recoding of the planning code to make the planner domain

[6] A description of some key domains from the competitions can be found in Section 4.2.1.

specific.

The 5th, 6th and 7th competitions all featured planning under uncertainty and machine learning, mentioned above. The 7th competition in 2011 also introduced a series of new domains extending the temporal complexity of benchmark problems. Required concurrency was introduced above, however this subtle complexity had not been featured in the IPC before 2011 as it had not been formalised. It is note worthy that older features must be revisited to correctly consider complexities and subtleties found later in the fields development. It may be that a more optimisation focussed approach to planning presented by the Unit Commitment problem and discussed in Chapters 5 and 6 may provide insights into numeric planning not previously considered.

Whilst each competition does present awards for the 'best' planners that participated, this is not main purpose of the competition. As mentioned above, the original purpose of the competition series was to facilitate comparisons, set up a framework for a common set of benchmark problems, and to measure the overall progress of Automated Planning as a field [151, p1]. Later the focus was to drive the direction of the community towards a certain goal, such as featuring real-world applications in IPC 4 [155], and plan quality in IPC 5 [153, p620].

There has been no consistent way of scoring the participants throughout competition iterations, and the difficulty of developing a fair scoring system was acknowledged in the original competition [151, p48]. Most competitions ranked planners based on plan quality, problem coverage, and solution time. Some organisers went to great lengths to base this on rigorous statistical analysis [130, Chp 4]. Others acknowledged it was more of an intuitive decision considering scaling properties alongside easily measurable quantities such as solution time and plan quality [142, p541]. Overall the awards are useful for identifying those systems which provide great performance over a wide range of problems. These are not intended to determine whether one system is categorically better than the other, simply to spur on competition and further the development of the field.

Multiple tracks within the IPCs has enabled a wider variety of problems to be modelled and solved using Automated Planning. These have branched into features

of planning which are beyond the scope of the topic of this thesis and are presented here for completeness only. It should also be noted that not all aspects of planning are covered by the competition. For example recent competitions focussed more on temporal than numeric domains and optimal planning has received much less attention than in early competitions.

Nevertheless, the competition still reflects the overall focus the community and provides an excellent opportunity to study the development of planning. What follows is a more detailed discussion of work relevant to modelling Unit Commitment with planning, specifically temporal and numeric planning with a focus on plan quality.

4.2 Planning For Optimisation

As discussed above, the IPC is an excellent set of benchmark problems. It highlights the state-of-the-art, community's interests and perceived challenges in the field of Automated Planning. In what follows a more detailed introduction to the topics of Optimal and Satisficing planning is presented. Each introduction is followed by an analysis of the performance of that class of planner throughout IPC iterations, taking specific interest in those domains similar to Unit Commitment.

Unit Commitment as a planning problem is a temporal-numeric domain with trivial proposition goals but an important plan metric. Full details are given in Chapters 5 an 6. As the IPC covers a wide range of problems, the published conclusions drawn from those competitions are based on the whole problem set. To understand the performance of planners on problems such as Unit Commitment, attention is restricted to a selection of domains outlined below. In some cases this analysis does not align with IPC competition conclusions. It becomes apparent that the modern direction planning has taken is not towards problems such as Unit Commitment, despite early interest and promise in this area.

4.2.1 Modelling Realistic Domains

Satellite Domain

IPC 3 in 2002 saw the introduction of the Satellite domain [130, p47], [155]. Inspired by, and developed through discussions with, NASA, the domain represents the real-world problem of satellites making observations, recording data and sending that data back to earth.

The two most relevant variants[7] are the, 'hard-numeric' and 'complex' variants.

In the hard-numeric variant, the propositional goals were very simple to achieve, but the problem metric was to minimise a linear combination of make-span minus the information sent back. The 'complex' domain added parameter dependent durations, not fixed durations for each action name. In each of these variants the goal predicates were trivially achieved so the planner must reason about plan quality rather than plan feasibility.

IPC 4 introduced a more realistic variant of the problem where communication between the Satellite and earth could only happen in certain windows of availability. This new variant is quite close to the Unit Commitment problem. It has a metric dependent on numeric fluents, set in a temporal domain including variable durations and requires support for Timed Initial Literals. Despite this domain being present in the early stages of the IPC, there are few other domains in the competition with such a combination of features.

The data for IPC 3 is no longer available so a re-analysis of the problems tackled is not possible. For IPC 4 the original problem files can be found online [156]. The complex variant in IPC 4 minimises make-span with the numerics only used to model constraints. This leaves the only domain close to Unit Commitment as the time-windows supplement of the complex Satellite Domain.

The problems for this domain range in size from 1 satellite, with 1 instrument operable in 3 modes, 6 directions in which to observe and 1 time window, to 10

[7]Each domain in the competition had multiple variations designed to be of increasing complexity allowing for comparison between new planners, established ones and an analysis of what aspects of a problem make it hard to solve.

satellites with 1 to 3 instruments each, each operable in 5 modes, with a total of 205 directions to observe and 44 windows to send information back in. This covers a large range of instances and the larger sizes represent a real world challenge.

Airport

Another realistic domain from IPC 4[8] is the Airport domain. The problem is to schedule the movements of planes from parked, to a specific runway segment, to taken-off. The most complex domain modelled all actions with durations, and had periods where runway segments could not be used, i.e. planes arriving as well as departing. There are no numerics but interesting temporal features including Timed Initial Literals.

The hardest instances were a half, and full model of Munich Airport (MUC), and make this one of a few available domains tackling a problem necessitated by industry. This is a very complex temporal problem for a planner, and what in reality would be the key objective, to minimise the total travel times of all planes not just the time of departure for the latest plane, could not be modelled in the competition version of PDDL 2.1 [155]. This highlights how simply modelling realistic *domains* as planning problems, before specifying and solving problem instances, is a complex challenge.

Below the results of this domain are detailed (see Sections 4.2.2 and 4.2.3), showing a distinct disparity between the first half of problem instances, deemed 'toy' models, and the latter half based on MUC. The 'toy' problems are up to 44 segments with 6 planes and 3 time windows, whereas the half MUC domains have 302 segments and 6 windows. This is a huge jump and some tougher 'toy' problems may have enabled an analysis of the limits the current planners could be pushed to. For reference, the toughest full MUC model contained 457 segments, 15 planes and 10 windows. This was proposed as a typical situation for MUC [142, p542].

[8] A description of each domain discussed from IPC 4 can be found in [155]

Rovers

The Rovers Domain, first seen in IPC 3 (2002) [130, p5], is another domain inspired by potential space applications. Here the planner is tasked with scheduling the movements of a planetary rover to collect samples, take observations and communicate that data back to a main base. The temporal aspects of the domain are scheduling movements and scientific tasks, and the numeric aspects are managing energy level and recharging. The optimisation is to minimise the total time taken to get all the observations.

This domain was also included in and extended in the 5th IPC in 2006 [153, p636] and included a temporal-numeric variant. The original files and results are available for this domain [157]. This variant does not require minimisation of a fluent based metric (only make-span) but does require the planner to reason about the interactions of fluents with time, as the `recharge` action's durations is dependent on current resources, the energy remaining, at the time of application.

Problems range in size from 1 rover, 4 waypoints and 2 objectives, to 14 rovers, 100 waypoints and 11 objectives (among other objects). Again this requires very good scaling from the planners to tackle all problems.

Power Supply Restoration (PSR) Problem

Another domain which highlights the challenge of modelling a real world problem as a planning problem is the Power Supply Restoration (PSR) domain from IPC 4 [155]. It attempts to model the problem of re-supplying parts of a distribution network in a power system after a fault has occurred. The domain includes actions to open and close switches and circuit-breakers, the problem goal being to plan a sequence of switches which results in a reconfigured network such that all sources are unaffected by the fault and a problem-specific set of lines have power restored to them.

The key disparities between this model and the full problem are an assumed full observability, numeric aspects such as capacity constraints and breakdown costs being ignored, and no optimisation. The modelling difficulty for the planner is that the pre-solve stage for a planner is to ground all the actions. Here the networks

modelled (ranging in scale such that "the largest instances are of the kind of size one typically encounters in the real world" [155, p5]) become intractable when grounded. Again, this shows that modelling real world domains presents many challenges to the community.

Travelling and Purchase Problem

The Travelling and Purchase Problem (TPP) only included in the 5th IPC [153, p640] [158] is a generalisation of the Travelling Salesman problem. In this domain the planner must schedule the movement of a buyer to various markets to buy various amounts pre-specified stock. The objective is to obtain all necessary stock and minimise a weighted sum of the time taken to obtain the stock and the price paid for that stock. This is a temporal and numeric problem with a clear optimisation goal.

This problem was included as it is has been proposed as a benchmark in other areas of computer science and is NP-Hard [153, p641], [158]. IPC 5 tackled problems ranging from a single depot, market truck and batch of goods to 10 markets, 4 depots, 10 trucks, and 25 types of goods. Again this shows a wide selection of problem sizes.

The Chemical Pathways Problem

The Pathways domain in IPC 5 [153, p638] [159] is inspired by modelling sequences of chemical reactions as actions. Each reaction consumes and produces different quantities of a substance. The goal is to produce a pre-specified amount of a selection of substances and the objective is to minimise a linear weighted sum of the amount of input substances and the total duration of the reactions. This problem is a realistic numerically focussed optimisation and it is unfortunate it was not included in later iterations of the competition to push performance on this type of domain.

Problem sizes ranged from 16 simple and 10 complex starting substances to 58 simple and 337 complex substances. The domain is based on the processes for representing the "Molecular Interaction Map of the Mammalian Cell Cycle Control and DNA Repair Systems" [153, p638]. It is unclear whether the problems themselves fully reflect the complexity of the molecular processes however the problems provide

a range of scales to test most planners.

Transport Domains

As discussed in Section 4.1.2 transportation and delivery domains are natural planning problems, able to be described purely propositionally or including a variety of interacting temporal and numeric variables and constraints. There has been a variant in IPC 1 [130, p45], 2 [129], 3 [130, p46], 5 [153, p642] [160], 6 [161] and 7 [162]. The goal has be propositional, i.e. purely deliver all packages, or to minimise either time taken, fuel spent, cost, or a weighted sum of all of these.

In IPC 5 the problem is purely temporal with preferences and constraint variants to stress the features of PDDL 3 introduced for that competition. In IPC 6 a temporal numeric variant is included which models varying package size, vehicle capacity, maximum fuel capacity and distance dependent fuel consumption to constrain the temporal minimisation to a more realistic model but is not part of the metric.

IPC 7 in 2011 introduced a variant called Nomystery, of note because there is an intrinsic, known, optimal solution to the problem produced by a "domain specific optimal solver" [162]. This allowed the organisers to push the initial amount of fuel to a level close to the minimum amount needed to solve the problem and test planners' ability to tackle highly constrained problems. This domain was non-temporal so will not be considered here as in practice there is usually much more capacity online in Unit Commitment than there is demand so it is not highly constrained in the same sense.

Openstacks

The Openstacks domain is another domain which has seen multiple entries in the competition. This is a combinatorial optimisation problem first introduced in IPC 5 [153, p635], [163] with a temporal-numeric variant. It is a common optimisation problem put forward as Constraint Programming benchmark and is NP-hard [153, p635], [163]. It involves scheduling a collection of machines to produce a selection of products to fulfil multiple orders. Only one machine can be used at once, and once production for one product from an order has begun, a 'stack' must be created.

This stack must remain open to 'store' the products for that order until production of all items in that order is complete.

The traditional optimisation goal is to schedule the production of each product so that all orders are fulfilled but the maximum number of stacks open at any point is minimised. The temporal aspect of the problem is to ensure that the correct orders are being completed simultaneously whilst the numeric aspect is to keep the number of stacks to a minimum.

The metric in the metric-time variant of the IPC 5 competition [157] is to minimise a linear weighted sum of the number of open stacks and total time of production. The IPC 6 [161] and 7 [164] variants include a numeric version with soft constraints so orders can be sent without fulfilling each requirement but this incurs a penalty. Production also has a cost, but the minimisation is to minimise make-span, not a numeric objective.

In the most relevant IPC 5 variant problems range from 10 orders of 10 product types to 50 orders and 50 product types.

The 6th and 7th International Planning Competitions

The Domains for IPC 6 are split into the following categories: sequential satisficing, sequential optimal, temporal satisficing and net benefit optimal[9]. Both sequential streams are for non-temporal domains and so are not considered here. Of the temporal satisficing domains the following have numeric variants: model train, woodworking, elevator, transport and openstacks (discussed above).

The model train domain requires the scheduling of trains around various segments of track. The numerics are to specify train sizes and location and are not the focus of optimisation.

The woodworking domain models the processing of wood scheduling jobs such as sanding and varnishing using a variety of machines. The numerics are to model

[9]For the Sixth International Planning Competition, termed IPC-2008, there does not appear to be a summary publication, as available for IPC 3 [130], 4 [142] and 5 [153]. The domain files and results for IPC-2008 are available online [161, 165]. The analysis that follows is based on my interpretation of those.

small problem specific features and are again not used for optimisation.

The elevators domain tasks a fleet of varying speed elevators with transporting passengers from origin floors to destination floors. The numerics are used to model capacity constraints with the objective to minimise the time taken. These have an impact on the optimisation problem which is to minimise total overall time. Larger elevators taking larger groups to close destinations will be faster than resending smaller elevators, however this is still a temporal optimisation problem not numeric. This domain was also included in IPC 7 however the numeric variant was dropped [166].

Domains of IPC 7 focussed heavily on parallelism and required concurrency[10] as discussed in Section 4.1.3 and no domains with numeric optimisation were included.

Temporal vs Numeric Optimisation

Many domains, such as the IPC 6 and 7 Openstacks variants, the IPC transport variants, IPC 7 elevator, trains and woodworking domains, can be grouped together in that the numeric features are not the objective part of the optimisation but the constraining part.

In what follows problems such as these shall be referred to as 'temporal optimisation' as the numeric fluents do not feature in the objective function. It is important to distinguish this type of problem from a 'numeric optimisation' where the fluents involved are directly included in the objective function to be minimised.

An example of purely temporal optimisation is the Airport domain, and later make-span minimising Satellite domain variants. An example of numeric optimisation would be the Chemical Pathways domain and 'complex' variants of the Satellite domain featuring in IPC 3.

In a planning problem, the heuristic used to guide the search is critical to the solution time and metric quality of the plans produced. A planner designed with temporal optimisation in mind will reason very differently to one not concerned with

[10] As with IPC 6, there does not appear to be a summary publication for IPC 7. A summary of domains is available [166] and the domain and problem files themselves can be found in [164].

time but with numeric optimisation.

It is important to note that the majority of the temporal-numeric domains encountered throughout the IPC are temporal optimisations and as such a planner's ability to perform numeric optimisation has not been explored by the community to the same extent. This implies the current collection of heuristics and search algorithms have not been developed with numeric optimisation in mind.

Problems such as Unit Commitment which are driven primarily by cost place much more emphasis on the value of the objective function rather than the time at which actions will be scheduled. For planners to be able to tackle real world problems featuring numeric optimisation, more research into domains such as those highlighted above may be required.

4.2.2 Optimal Planners

One key planning system which produced plans that were provably optimal with respect to make-span was Graphplan [167]. Its problem representation and search method inspired the popular Relaxed Planning Graph heuristic and many planners referenced throughout this review. Later, Haslum and Geffner generalised the heuristic in Graphplan, detailing a family of admissible heuristics calculating increasingly accurate lower bounds at the cost of increased computation time [168]. The functions are denoted $h^2, h^3, \ldots h^m$. h^2 was the heuristic found in Graphplan, and $h*$ is often used to denote this family of heuristics. Full details are beyond the scope of this project and can be found in [168].

In this sense a planner could be run in different settings to either provide more accurate searching or provide faster but less accurate solutions. This heuristic was used to estimate the cost to go within an Iterative Deepening A* search (IDA*) so their planner, HSPr*, could find provably optimal plans for non-temporal, non-numeric planning problems.

Variants of this HSP family of planners were entered into most IPC events. Two optimal variants were entered into IPC 4; HSP*_a and TP4-04. They are of particular note in that they are the only optimal planners the author is aware of to handle both temporal and numeric domains. The two variants are identical apart from the

heuristic. TP4 uses the h^2 variant of the $h*$ heuristic and HSP*$_a$ uses a variant which calculates a (still admissible) relaxed version of h^m where $m > 2$.

A detailed analysis of the domains from IPC 4 was conducted by Haslum shortly after the competition [169] with a bug fix version of HSP*$_a$ and showed that the relaxed heuristic is not always beneficial but in some domains can provide a large improvement. Unfortunately both versions were still not competitive on non-numeric temporal domains where other systems were competing.

Other attempts at creating optimal planners included translating the planning problem into other forms where known algorithms could find optimal solutions. The first non-specialised approach to be widely successful was the satisfiability (SAT) approach, when Kautz and Selman introduced SATPLAN in 1992 [149].

SAT problems are assigning boolean variables in boolean formulae values to make the overriding expression true. SAT solvers are a very active area of research and so translating planning problems into SAT problems yields a whole field of research to tackle them with.

After the success of SATPLAN, the authors developed Blackbox [170] which automatically translated a STRIPS planning problem to a SAT problem. An updated version of Blackbox, SATPLAN-2004, was later entered into the pure-STRIPS part of IPC 4 and was awarded first place in the optimal stream [142].

Non-temporal non-numeric planning problems can be seen to naturally have a parallel in SAT problems, however there appears to be no work formulating temporal or numeric planning problems as SAT problems.

Another example of a planner using another field of research is CPT [145], which translates the planning problem into a Constraint Satisfaction problem informed by the h^2 heuristic, which in turn is solved. Multiple iterations of CPT have been entered into the IPC since IPC 4. CPT2 in IPC 5 introduced a better pruning technique during search [153, p646], CPT 3 and 4 continue this progress however lacked direct competitors as the optimal temporal track was removed from the competition due to lack of entries [146].

In an attempt to assess the competitiveness of the CPT family of planners, Vidal performed a review of CPT4 against time-of-publication state-of-the-art SAT based

parallel[11] planners [146]. CPT4 comes 3rd in terms of problem coverage behind SASE and the latest version of SATPLAN, however is generally comparable in terms of computation time. As there are no competitors for optimal temporal planning CPT 4 is only compared with CPT 3 and shown to have favourable coverage and performance.

After IPC 4 the optimal stream reduced to non-temporal non-numeric domains and to my best knowledge no further numeric-temporal versions of the HSP family of planners able to tackle temporal numeric domains have been developed. This indicates that there is perhaps not much community interest in optimal planning as other areas of mathematics and computer science indicate better performance with regards to provably optimality.

Performance of Optimal Planners within the IPC

The relative performance and coverage of optimal planners in IPC 3 and 4 is summarised. Due to the aforementioned reduction in participation in the temporal optimal stream of IPC below these were the only competitions available for analysis.

IPC 3 had optimal planners competing alongside satisficing planners: Semsyn [171] and TYPSYS [172], Graphplan based optimal planners, and TP4 [173]. Semsyn in this iteration could handle STRIPS and Numeric domains, TYPSYS could handle STRIPS and SimpleTime domains (where durations are not situation dependent but fixed per action), and TP4 could handle Numeric, SimpleTime, Time and Complex (where durations are situation dependent and there are other numeric resources as well as time) domains.

In the two semi-realistic domains of IPC 3 (Satellite and Rovers) only TP4 could compete. Of these problem instances, coverage was so low[12] that the statistical

[11] Parallel planners here refer to a subset of temporal planning problems which are not as expressive as other forms. Actions can be ran alongside others (i.e. 3 actions can run at time step 1) and actions can take multiple time steps, however start and end preconditions cannot be expressed. See [132, 146, 167] for examples of parallel planning.

[12] Individual results files from IPC 3 are no longer available however the total coverage for Semsyn,

analysis performed in [130] ranked the planner so low we cannot assess the quality of the satisficing planners against the results of TP4. This demonstrates why optimal planners were separated from satisficing planners in later competitions.

The results from IPC 3 lead to the suggested dismissal of Graphplan based planning [130, p51]. However, for the majority of the previous IPC domains, it was demonstrated that finding a provably optimal solution is a different complexity to finding a feasible solution; the latter being polynomial and the former being NP-Complete [142, p522]. With this in mind the 2004 competition saw optimal planners moved to a second stream allowing for a clearer analysis of how optimal planners perform relative to each other.

IPC 4 contained many interesting domains based on realistic applications. An updated version of Semsyn was entered into IPC 4 this time tackling numeric domains. The success ratio increased from 8% in IPC 3 [130, p8] to 40% in IPC 4 [142, p538] demonstrating that there is perhaps unrealised potential within optimal temporal planners if they were developed to the same level as satisficing systems.

In the Airport domain non-temporal results are good especially from SATPLAN. All 'toy' instances are solved by most planners but as scale increases the optimal planners suffer, with SATPLAN solving 7 of the half MUC systems and 3 MUC scale of problems and all others only solving 3 instances [142, p543].

For the more realistic temporal variant, CPT could solve all of the toy instances. TP4 could solve 12 of the toy variants, with the problem instances with more planes proving more problematic than larger airports[13]. TP4 could also solve the simplest of half MUC scale problems with only 1 plane.

For the satisficing planners tackling this domain, in all instances either LPG tuned for quality or LPG as entered in IPC 3 tuned for quality gave the best metric, with the better metric coming from the version which happened to take the longest

TYPSYS and TP4 was 8%, 12% and 13% respectively. In comparison, the lowest coverage for a satisficing planner was 49% [130, p8].

[13] HSP^*_a's performance was as good as but never better than TP4 and as it had less coverage and the competition version contained bugs in its implementation [169, p249] so has been omitted from discussion

time to solve. As LPG for IPC 4 has better coverage, that is used as a comparison.

Table 4.3 shows the metric values attained by CPT[14] and LPG. It also shows the time taken to achieve that value. In almost all the cases where the satisficing planner, LPG, finds the optimum value, surprisingly, CPT is faster at finding this value. However, in those instances when CPT finds an optimal solution and LPG doesn't, LPG is much faster. Problems 16, 17 and 18 see LPG find a solution that is 4.3%, 7.3%, 8.5% worse than CPT but at a time that is 7.9, 10.6 and 16 times faster, taking seconds instead of minutes. LPG on domain 20, with the most planes of a toy problem, yields a solution 43% higher than the optimal but CPT takes 20 times longer.

Results are therefore mixed from this domain. Whilst for small instances CPT is genuinely the better option, clearly outperforming others, its reduced coverage and inability to tackle even the 1 plane variants of the half MUC and MUC domains weaken its impact on real world planning problems. With all other optimal planners being outperformed by satisficing ones it is understandable that community interest diminished.

Non of the optimal planners supported the Timed Initial Literal language constructs, so none could attempt the most realistic version with windows of availability.

The Power Supply Restoration domain is non-temporal and non-numeric and whilst coverage is very good on small instances across the selection of Optimal Planners [142, p557], again they cannot handle the language constructs of the more realistic problems.

In the Satellite domain, only the 8 smallest instances of temporal problems could be solved by CPT, whilst HSP^*_a and TP4 could only solve 4 of the first 5. For the temporal-numeric problems CPT could not compete as it does not handle numerics. HSP^*_a and TP4 could only solve the same 4 out of 5 problems.[15]

Despite each optimal planner generating equal solutions, they were bettered by

[14]In the instances TP4 could solve it achieved the same value but took longer to solve so it has been omitted from discussion.

[15]It would be good to compare to the IPC 3 results however these files are no longer available online.

	1	2	3	4	5	6	7	8	9	10
CPT	64	185	200	127	227	232	232	394	402	126
LPG	64	185	200	127	227	232	232	394	402	126
%										
CPT	0.03	0.03	0.19	0.12	0.26	2.71	2.5	20.45	34.79	0.1
LPG	2	2	2	4	4	6	6	10	11	5
CPT vs. LPG	0.02	0.02	0.10	0.03	0.07	0.45	0.42	2.05	3.16	0.02

	11	12	13	14	15	16	17	18	19	20
CPT	228	228	230	390	262	393	399	435	413	435
LPG	228	232	230	390	262	410	428	472	413	625
%		**1.75**				**4.33**	**7.27**	**8.51**		**43.68**
CPT	0.4	5.3	5.2	36.5	34	110.4	180.4	399.6	208	543.3
LPG	5	7	7	11	8	14	17	24	89	27
CPT vs. LPG	0.08	0.76	0.74	3.32	4.25	**7.89**	**10.61**	**16.65**	2.34	20.12

Table 4.3: Table showing the metric value attained and CPU secs taken to attain it for the optimal planner CPT and satisficing planner LPG in the Temporal variant of the Airport domain without Timed Initial Literals for windows of availability. We see that for simple problems CPT is genuinely the better planner finding the optimal solution in less time, but for more complex problems (14, 15, 19) it is slower to find the optimal. When LPG does not find the optimal it gets close within a much shorter time frame.

	1	2	3	4	5	6	7	8
CPT	135.5	156.311	65.198	122.24	105.26	64.824	60.202	74.024
TP4	135.5	156.311	65.198		105.26			
LPG (3)	129.58	152.28	55.25	115.24	97.72	67.1	59.84	83.3
LPG (4)	129.5	187	55	127.83	121.08	120.54	77.9	115.13

(a) Results for the Temporal Domain

	1	2	3	4	5
TP4	135.5	156.13	65.2		105.26
LPG (3)	129.5	152.3	55.3	115.2	97.72
LPG (4)	133.97	181.05	60.88	145.47	125.05

(b) Results for the Temporal-Numeric Domain

Table 4.4: The metric values and CPU secs for a selection of satisficing planners (LPG as in IPC 4 and LPG as in IPC 3) and optimal planners (CPT and TP4) on the temporal and temporal-numeric variants of the Satellite domain from IPC 4. Results show the poor relative performance of the optimal planners demonstrating the complexity of supporting all features the syntax is capable of.

one or both of the LPG variants in 6 out of the 8 temporal, and all 4 of the temporal-numeric, variants. Table 4.4 shows these results.

This highlights that some planners can only handle a subset of the language constructs, and may not handle them correctly. CPT requires a simplification to the full complexity of durative actions specifiable by the PDDL 2.1. These simplifications lead to situations where incorrect solutions are found, and highlights that whilst the algorithms and heuristics may be theoretically optimal the implementations of the model may not fully reflect that, and of course the model itself may have inherent simplifications. As always when considering optimisation problems one must be aware that the solution given is only optimal under certain assumptions and restrictions.

The same occurs for Semsyn in the numeric non-temporal variant of Satellite, the only optimal planner to tackle it. Conference organisers were unsure why, but results demonstrated that LPG produced solutions of a higher quality [142, p562].

In IPC 5 there were many interesting Temporal-Numeric Domains (§4.2.1) however only 1 of the 6 optimal planners entered could handle temporal constraints (an updated version of CPT). None of the planners entered could handle both temporal and numeric domains [153, p645], so there are no results to compare here. In IPC 6 and 7 the optimal track only considered sequential problems, i.e. problems with no temporal durations, due to lack of entries for the optimal-temporal track [146].

Conclusions on Optimal Planning

From this brief overview of optimal planners it should be clear that the community has taken them in a direction away from the kind of problems this project is focussing on. Early optimal planners competing in IPC 3 and 4 compared favourably in the time taken to find the optimal solutions that could be found on small problem instances. However, satisficing planners still found those solutions and as these solution times were generally small anyway this gain on its own has little impact. The significantly reduced coverage of optimal planners compared to the best satisficing planners, especially on realistic domains and problem instances, gives little indication that pursuing them further would have been fruitful.

That the community has restricted the scope of optimal planning competitions to sequential problems in later IPCs shows planning has not been pushed towards provably optimal numeric optimisation. That is not to say plan quality has been ignored, indeed Section 4.2.3 highlights the many systems where plan quality has been a major concern.

It can also be seen from tracking the benchmark problems in the IPC that it is very hard to support many of the language constructs, such as Timed Initial Literals and the full complexity of concurrent durative actions. There also appears to have been little work on incorporating numerics into the optimal planning framework. TP4 was the only such optimal planner which competed, and it compared favourably with neither the other optimal planners or the best satisficing planners.

Perhaps interest has wained in optimal planning, because even in pure STRIPS problems in many cases the problems can be translated into a form where planning based approaches perform worse than pre-existing methods, as demonstrated by CPT's performance against SAT based planning. This indicates than planning's real strength lies in tackling domains that are difficult to model in other forms, such as concurrent temporal-numeric domains.

Finally, there is an intrinsic difficulty in applying the h* heuristic, which could be developed to include temporal and numeric goals to a problem such as Unit Commitment. When formulated as a planning problem (see Chapter 5.2.3) Unit Commitment has a dummy goal to ensure the planner creates a plan for the whole horizon. This is very similar to the hard-numeric and complex-time problem instances in the Satellite domain which contained language constructs not supported by any optimal planner entered into any IPC. The h* heuristic is calculated based on the cost of achieving sets of goals and their preconditions. Even if there were accepted admissible ways of incorporating the necessary numerics to model Unit Commitment as a planning problem into the h* heuristic, it is probable that there are too few goals in the natural formulation for the relaxation to be effective. The requirement to perform actions in a Unit Commitment domain is in response to Timed Initial Fluents and reasoning about those in an admissible way within a Planning framework is work that appears not to have been done.

To work around this problem and put Unit Commitment in a form more suited to the h^* heuristic one could introduce predicates `served_d`i, implying demand period i has been satisfied, the achievement of which forms the problem goal. This however would require actions to set these predicates to true. A formulation of such actions which does not result in a huge increase in the number of actions, thereby removing the idea that there are fewer actions and decision points in a planning problem than a MIP model, eludes the author.

As optimal planners have only demonstrated performance on small problem instances and less realistic sequential domain variants, with demonstrable fallibility within those, attention is now restricted to satisficing planners.

4.2.3 Satisficing Planning

Satisficing planning has been very popular due to its ability to tackle more realistic domains by handling the full complexity of language constructs. By definition, these planners give no guarantee of optimality as the heuristics are not admissible and algorithms not optimal. Where the current industry practices (see Chapter 2.3) include a method which does provide an upper bound gap to optimality, and gets very close to that optimal solution, it is important that any method which does not provide a provably optimal bound is rigorously compared to its competition.

Many satisficing planners make use of a Relaxed Planning Graph (discussed in Section 4.1.2), first presented in [167]. They differ through the heuristic computed using this graph and the search algorithm implemented. One heuristic is to estimate the number of actions until the goal is achieved by propagating only the add effects of actions from the current state until all goal predicates are achieved. This estimation was first presented independently by [174, 175]. This is an attempt to find the plan with either the lowest 'make-span'. This is an extremely popular heuristic and has been employed by numerous planners since publication [122, 133, 176, 177].

It is complex to incorporate temporal and numeric properties into this planning graph, but as the formulation has proven to be so popular it is work that has been done, indeed planners must tackle these issues if they are to concern themselves with realistic domains as discussed in §4.2.1.

Hoffmann extended the idea of ignoring delete lists to incorporate numeric fluents into the relaxed plan framework, enabling the same basic algorithm from FF [122] to be applied to numeric problems. His planner, Metric-FF [176], first competed in IPC 3 in 2002[16] and was very successful and influential [130] with 5 out of the 12 satisficing planners competing in IPC 4 being extensions of FF or its heuristic [142].

His algorithm enabled a planner to attempt to find a plan with the shortest make-span whilst respecting numeric constraints. Metric-FF also supported Pddl 2.1's semantic ability to express a metric representing plan quality. This specifies that the resultant plan should minimise a given numeric quantity[17]. There are various restrictions on the objective function including linearity and more crucially, the ability for the objective function to "*be transformed ... into an additive action cost minimisation*" [176, p318]. This means that each action is assigned a cost based on the numeric consequences of the action's effects to the objective function. This means that to the planner instead of an action 'costing' 1 step and therefore simply adding 1 to the make-span, it costs an amount dependent on the terms of the linear objective function affected and how they are affected.

The algorithm then proceeds as normal with the relaxed planning graph being formed as before but now the relaxed plan with the lowest make-span is the relaxed plan with the lowest expected increase to the objective function. In Metric-FF this is the state which is expanded next in the search, the idea being that always choosing the action which will appear to give the lowest metric cost will produce a high quality plan.

This heuristic is not admissible and so there is no guarantee of optimality or of finding a gap to optimality. The relaxed plan is formed by ignoring negative effects, the reinstatement of which, if necessary, could greatly add to the cost of the plan. Thus having the lowest relaxed cost does not guarantee a low actual cost.

Another early planner that strives for plan quality using this additive action cost minimisation approach is LPG [144]. LPG differs from FF and Metric-FF

[16]The term Metric-FF was introduced by Hoffman in [176] after the competition however the FF variant in IPC 3 was acknowledged to be Metric-FF.

[17]Maximisation is supported but treated by multiplying the objective by -1 and minimising.

as it uses Partial Order Planning. It uses a stochastic local search, which could be a disadvantage for Unit Commitment as solution variability will be an issue (as discussed in Chapter 3). Conversely, an advantage of LPG and partial order planning for Unit Commitment might be that it could be warm started with a plan from a priority list or other very fast method. LPG and its later incarnations were very successful, particularly in IPC 4 where, as discussed below, they produced many plans of optimal quality.

Temporal Planning

Like numeric planning, temporal planning has many subtleties and complexities. As mentioned above, more complex features of temporal planning, such as required concurrency and continuous change (see below), have developed as the field matured.

The idea of Parallel Planning could be seen as the first representations of temporal planning. Knoblock was one of the first researchers to formalise parallel planning, defining it to be a plan where independent actions could be ran in parallel assuming enough workers could be found [178]. Consider a mechanic who has to mend a vehicle. The tasks can be grouped to fetching the necessary tools, using them to fix various car parts and returning the tools. Whilst fixing some parts may be predicated on the fixing of others, the fetching and returning of tools can be all be done in parallel assuming enough workers. The plan would thus be to fetch the tools in the first step and return all the tools in the last step, even if each of those time steps contained multiple actions.

This differs from the previously described notion of planning where one action is "indivisible and uninterruptible", defined as Atomic Planning [178, p98]. Early planners such Graphplan [167] and Blackbox [170] plan in this way. It has a natural affinity with partial order planning, where formal ordering of the actions is not finalised until search is complete.

By utilising the affinity with partial order planning more expressive temporal solvers were developed. IxTeT is a partial order planner developed in 1994 [179]. Being one of the earlier temporal planners it had to define its own input language and had to balance how expressive it was with the efficiency of search. As such it was

not so competitive when modern planners and the IPC competitions were introduced [131, p34]. Driven by interest in space applications, HSTS, EUROPA and Aspen are further examples of early temporal planners with specific input languages [131, p34]. These planners required domain specific user input to achieve efficient search, rather than being generic solvers.

All were acknowledged to have rich representation languages for the types of search performed by the planners. PDDL 2.1 was introduced for the 3rd IPC in 2002 [127, 130] as it was hoped a common language would expedite the development of temporal planners and widen the scope of the field[18]. The durative action formulation discussed in Section 4.1.2 allowed for start, end and overall preconditions and start and end effects making it more expressive than parallel planning and combining some features of the above early temporal planners.

Many temporal planners have been developed to support this PDDL variant, including partial-order planners [143], more advanced planning-graph planners extending Graphplan and Blackbox [144], and Optimal Planners such as TP4 and HSP^*_a discussed above among others.

Despite the expressive abilities of PDDL 2.1, it is was later found that most of these temporal planners and temporal domains from IPC competitions prior to IPC 6 were incomplete, and did not correctly handle all classes of temporal problems. Cushing *et al.* [132] formally defined the notion of required concurrency and noted that most existing planners could only handle temporal problems which could be solved by decomposing the problem into a non-temporal problem (i.e. what actions to perform) and then scheduling them once an order has been found (i.e. when to perform those actions).

LPGP was one of the first planners to correctly handle required concurrency, separating the logical structuring (what actions to perform) and the scheduling (when

[18]There remains ongoing discussions throughout the community about the best way to express different kinds of temporal problems. A debate on the merits of the different proposals for expressing temporal problems is beyond the scope of this work. PDDL 2.1 was chosen for this project as it was the input language used by the only planner I was aware of able to handle all the requirements for a planning formation of Unit Commitment (see Section 4.3.1 for further details on this).

to perform those actions) but reasoning about both throughout search [180]. It does so by using a Graphplan based search for the logical structuring with tighter mutex relations and maintaining a collection of linear constraints representing temporal relations between actions.

Another approach handling required concurrency is CRIKEY [131]. The approach used here is a forward chaining state space search where temporal relations and maintained in a Simple Temporal Network, a formalised set of constraints similar to that used in LPGP. It is discussed in more detail in Chapter 6.3. CRIKEY is of note as it developed into very competent planners, POPF and COLIN, which are proposed as suitable planners for tackling Unit Commitment in Chapter 6.

Temporal-Numeric Planning

One reason for the success of the above planners was their ability to handle both temporal and numeric structures. There are a class of temporal problems where numeric change is dependent on temporal structures, known as problems with continuous processes. In PDDL they are represented by the '`#t`' construct which implies the increase or decrease is not an instantaneous change to the value of the fluent, but an instantaneous change to the rate of change of the fluent. All numeric effects discussed so far are assumed to have an instantaneous effect. However for many realistic problems the numeric effect should be continuous over a period of time.

This presents a challenge for planning, especially methods relying on decomposing the logical and scheduling components of a temporal problem mentioned above. A sequence of actions which holds sequentially may not be valid in the presence of continuous numerical processes, where numeric conditions may become violated during an action's execution. To ensure numerical validity throughout the temporal duration of the plan, the numerical constraints and conditions must be captured alongside the temporal constraints.

PDDL+, an extension of PDDL 2, allowed exogenous processes and events to instigate continuous change. These occurred outside of the action framework, i.e. their occurrence was predetermined and not scheduled by the planner. It is therefore possible to model complex temporal-numeric real world processes in PDDL. The

Chemical Batch Processing Problem was proposed as a benchmark problem for PDDL+ [181]. It models the production of chemicals, saline solutions in this instance, and contains a mix of discrete and continuous processes with complex non-linear dynamics.

UPMurphi [147] is the planner proposed to solve the Chemical Batch Processing Problem. UPMurphi was used to test initial models of Unit Commitment as a planning problem, an in-depth description of the planner and implementation can be found in Chapter 5.3. UPMURPHI is able to solve planning problems with continuous processes using the discretise and validate approach. Rather than directly handling the continuous processes, the plan is validated at iteratively finer time granulations to ensure the current dynamics do not validate the original problem constraints.

An earlier, more traditional planning approach was Kongming [182]. Developed in 2008, it is an extension of a Graphplan approach containing a mathematical programming sub-solver to handle continuous processes. The authors demonstrate the capabilities of their system on an Autonomous Underwater Vehicle (AUV). The limiting factor of the Graphplan approach is that the planning graph becomes intractable when the plan size required grows too large. The online system of the AUV avoids this issue as only a few actions are required to be planned before replanning will occur.

COLIN, a temporal numeric planner capable of handling continuous linear processes [133], manages temporal and numeric actions by replacing the set of temporal constraints in CRIKEY (discussed above) with an LP incorporating both temporal and numeric constraints. When a discrete change occurs to the rate of change of a numeric variable the LP constraints are updated[19]. In this way multiple actions can apply continuous changes to a single fluent, overcoming a major weakness of early temporal-numeric planners such as Zeno and OPTOP [133, p20]. Empirically it has demonstrated much stronger performance than Kongming. It is more effective than UPMURPHI as it has an inbuilt heuristic, whereas UPMurphi requires a heuristic to be built into the planning model, which whilst offering flexibility, makes the planner

less generically applicable.

Unit Commitment is a problem which requires continuous numeric change in the ramping of units. Changes to variables, specifically the total supply, is instigated by multiple actions, namely the ramping up and down of multiple units. COLIN is the most capable planner demonstrated in the literature that has these requirements. Since their original publications, the development branches of POPF and COLIN have merged as they contained much duplicate reasoning. For this reason, the latest version of POPF, supplied by Maria Fox and Derek Long, was to be used for testing. Full details of Unit Commitment as Planning using these planners can be found in Chapters 5 and 6.

Performance of Satisficing Planners within IPC

IPC 3 in 2002 was the first time planners competed in a metric setting. METRIC-FF and LPG were two stand out planners from that competition with METRIC-FF consistently outperforming other entrants in solution time [130, p21] and LPG consistently outperforming other entrants in solution quality [130, p23]. In all of the STRIPS categories LPG was shown to be equal second for solution time, on Numeric domains second only to FF and in simple-time and temporal domains it was the fastest.

The analysis showed that LPG was an extremely strong planner, receiving an overall award for the best automated planner. FF also garnered a strong reputation and an award for the best performance in the numeric domains [130, p9]. The full data set for IPC 3 is no longer available and so a re-analysis restrict to just the realistic domains considered here is not possible.

Results from IPC 3 on the 'hard-numeric' and 'complex' variants were disappointing with only the hand-coded planners performing (both equally) well [130, p10,11]. The two other planners capable of solving these variants, MIPS and FF, solved all instances (as they were propositionally trivial) but with poor plan quality.

[19] A full description of the LP constructed by COLIN to ensure temporal and numeric validity can be found in [133, Chp 8].

Although the results for IPC 3 are unavailable, LPG from IPC 3 remained a satisficing entrant in IPC 4 for which the entire data set is available [156,183]. LPG was deemed an overall second for speed, and first for quality, in IPC 3. In IPC 4 LPG-TD was awarded two first places for the two types of temporal domains, implying it retained its performance well.

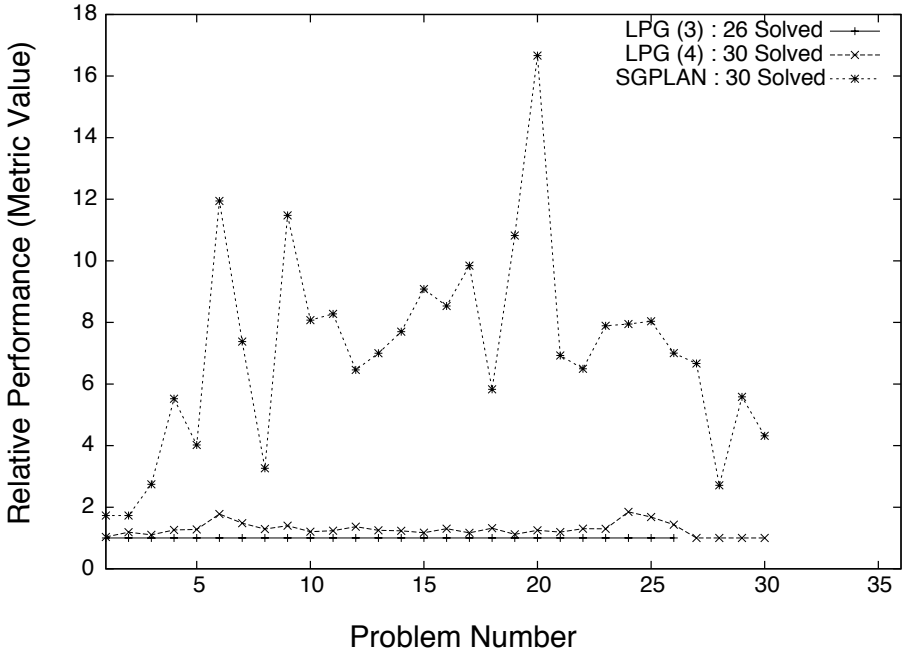
Coverage and performance for the complex variants of the Satellite domain improved slightly from IPC 3 to 4, as can be seen from the results in Figure 4.3. This is in contrast to other domains such as the Airport domain discussed below which saw a great improvement. This implies that the numeric side of planning has received less attention than other aspects. This appears to be a common theme throughout IPC iterations.

The planner SGPLAN appears in all IPC iterations from IPC 4 onwards showing good performance across many domains. It searches for a plan by decomposing the planning problems into smaller sub-problems [184]. Any inconsistencies are resolved as the sub-plans are amalgamated.

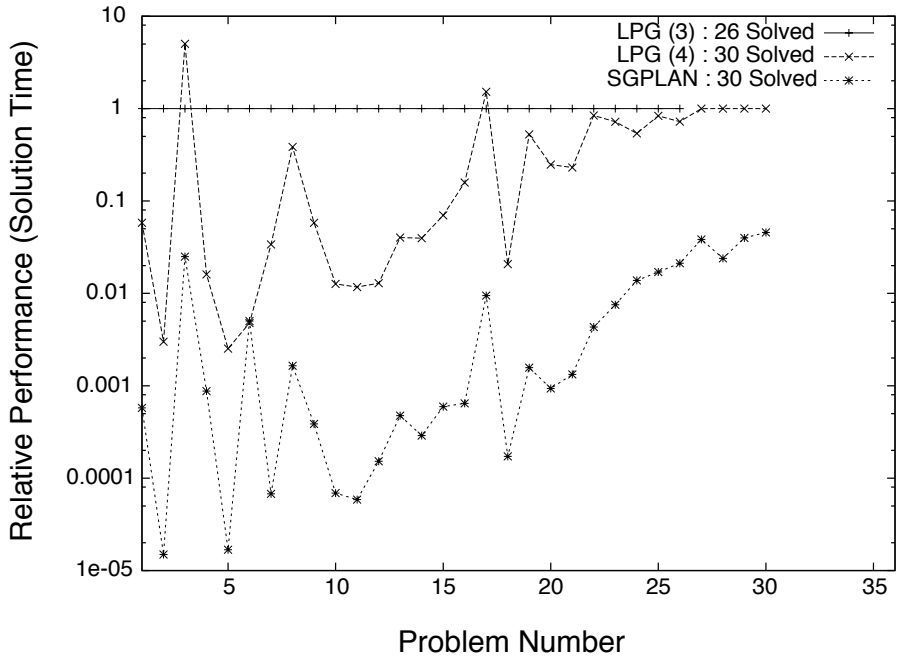
The make-span data for SGPLAN, was not analysed in [142], only the solution time and number of actions, which is the default for SGPLAN to minimise. To give a fair comparison on the realistic domains considered here, this data has been compiled from the raw results files [183] and can be seen alongside the LPG-TD and LPG data in the two graphs show in Figure 4.3.

What we see here is that whilst the solution time is orders of magnitude shorter for SGPLAN the make-span of the plan, which was the goal metric in the IPC 4 complex variant, is much longer so the quality of the resultant *temporal* plan is worse.

Figure 4.4 shows the data for the most realistic variant, which includes a numeric metric and windows of availability for communication. It reveals SGPLAN again has increased coverage and solution speed over the other competitors. One can analyse the plans generated by SGPLAN to see that this time plan quality is comparable and occasionally better than LPG. Also, LPG can be configured for speed and then only lags slightly behind SGPLAN. It is important to understand in what configuration the planner was run before assessing the over all performance.

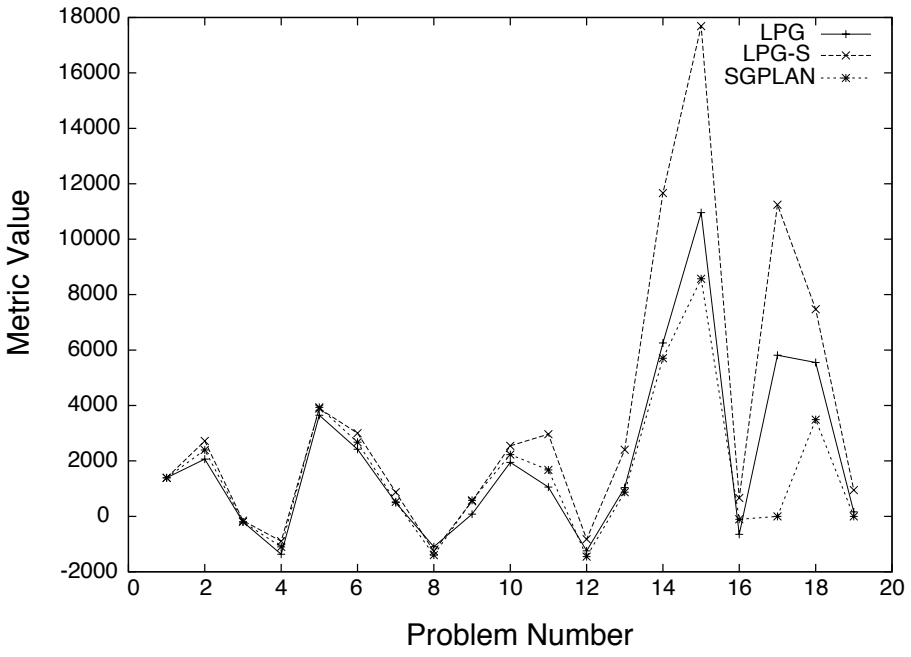


(a)

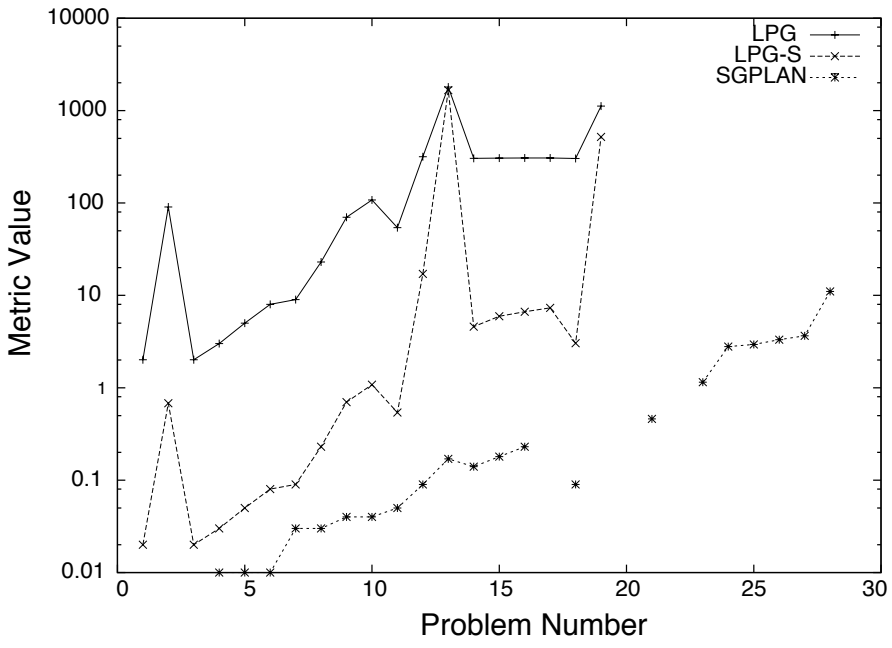


(b)

Figure 4.3: Relative performance of LPG in IPC 4 and SGPLAN against LPG in IPC 3. Lower values are better. This demonstrates LPG making a small improvement and SGPLAN solving orders of magnitude faster but dropping solution quality up to an order of magnitude relative to the two LPG variants.



(a)



(b)

Figure 4.4: Actual performance of LPG in IPC 4 against SGPLAN. LPG 4 has a huge disadvantage with regards to time, but can be configured differently (LPG-S) to achieve times close to SGPLAN, with comparable solution quality. In this domain variant the metric appears to be have been considered by SGPLAN, with plan quality comparable to, and sometimes better than, LPG 4.

It should be noted that in the non-numeric time windows problem the organisers state that SGPLAN has a "clear advantage" [142, p562] over LPG-TD. This indicates that much emphasis was put on problem coverage, solution time, and temporal problems, rather than the combined temporal-numeric domains with numeric optimisation goals - the type of domain required to model Unit Commitment.

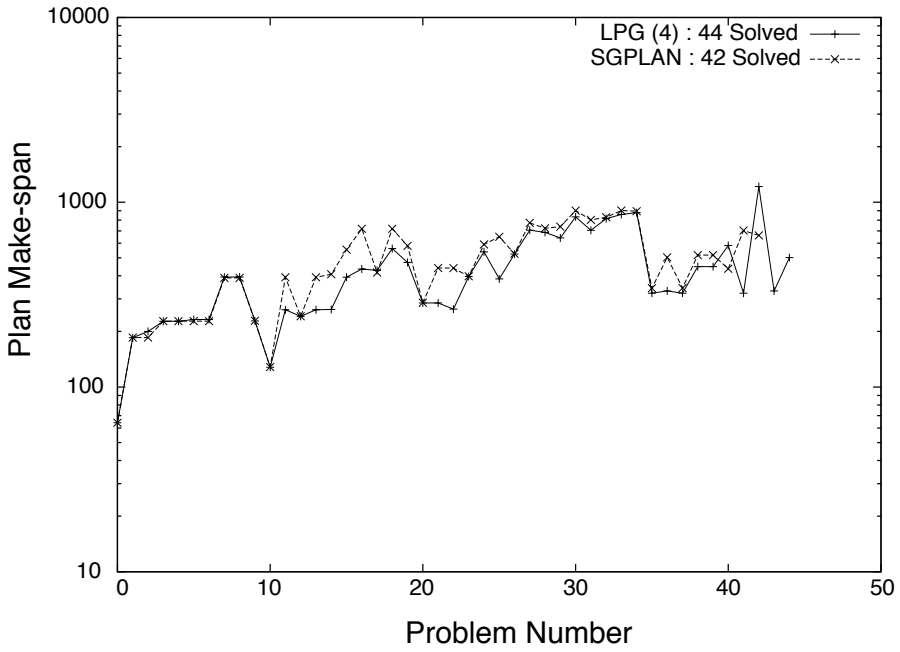
The Airport domain was highlighted because it featured real-world domains necessitated by industry. The key question is whether those domains were successfully tackled. Graphs in Figure 4.5 show that coverage is really very good. Solutions up to problem 44 (457 segments, 5 planes, 10 windows) are found within the 30 minute time limit of the competition. SGPLAN and LPG-TD had by far the best coverage for this domain variant [142, p545] and so are the only ones depicted. The strong and comparable performance on realistic domains by both planners with quite different search strategies highlights the strength of temporal planners.

The interesting features of the Rovers domain were the action durations dependent on numeric fluents. Problems ranged from 1 rover, 4 waypoints and 2 objectives, to 14 rovers, 100 waypoints and 11 objectives (among other objects). This was ambitious coverage and performance in IPC 3 cannot be easily assessed due to the lack of raw data already mentioned.

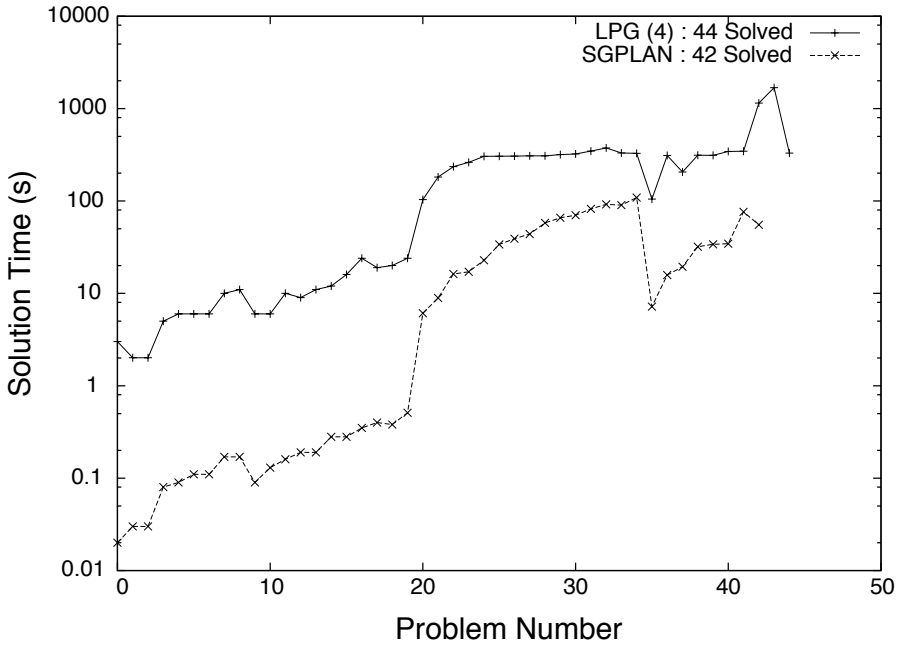
In the IPC 5 Rovers Domain, SGPLAN gave the best coverage but for a price in solution quality [185]. For problem numbers 20 and upwards solution times strayed above minutes and SGPLAN was the only planner able to find a solution. For quality, another planner, YochanPlan, appears the best but coverage and solution time were both poor. Given Unit Commitment shares some of the key features of this domain these results may be discouraging for a full planning model for Unit Commitment, such as the first model presented in Chapter 5.

The three other domains with temporal-numeric variants in IPC 5, the last competition to feature such domains, were the Travelling and Purchase Problem (TPP), the Chemical Pathways Problem and the Openstacks problem.

The Metric-Time variant of TPP only saw SGPLAN giving competitive performance. MIPS solved 6 instances and no others competed, where as SGPLAN solved 39 of 40 instances. No graphs / tables are given as there are no comparisons to other



(a)



(b)

Figure 4.5: Results of the two best performing satisificing planners in IPC 4, LPG and SGPLAN, for the Airport Domains. In contrast to the temporal-numeric domain of Satellites the results for this domain remain close even for the more complex problems (higher problem number) implying that technologies for temporal planning have advanced further.

planners to be made. The goal of SGPLAN is always to minimise make-span and perhaps coverage is so good because it is not actually performing the optimisation, unlike MIPS which is, and is therefore getting stuck.

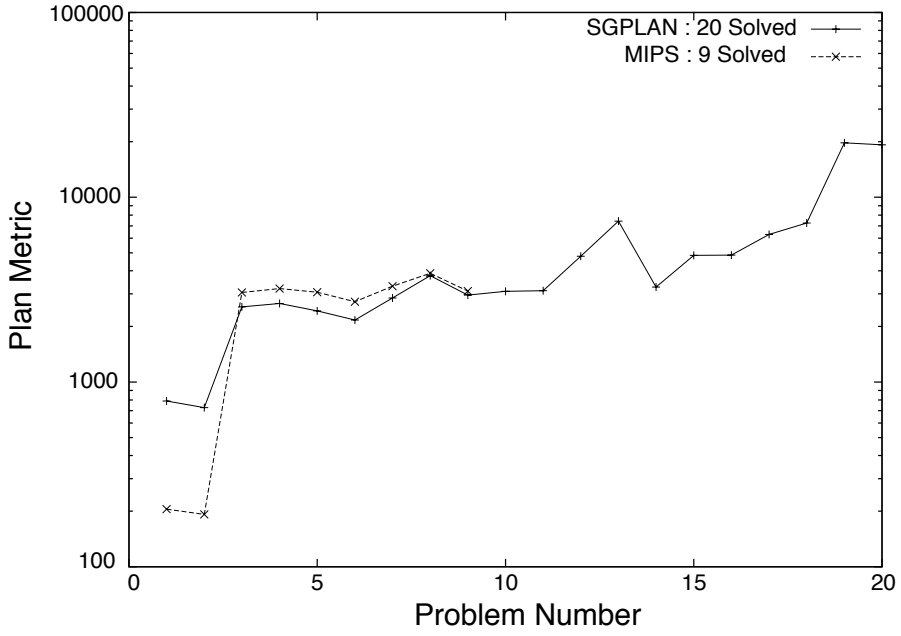
The same accusation can be made of SGPLAN in the Pathways Metric-Time variant. SGPLAN finds valid plans for all problem instances however plan quality is worse than MIPS for the 1 instance MIPS solves.

The performance in the Metric-Time variant of Openstacks however implies that plan quality by SGPLAN is respected. The graphs in Figure 4.6 shows that metric quality for SGPLAN on the 9 instances MIPS also solved was better. It also showed significantly faster solution times by 2 orders of magnitude. A more detailed analysis of these two domains and others would be required to understand under what circumstances SGPLAN performs well.

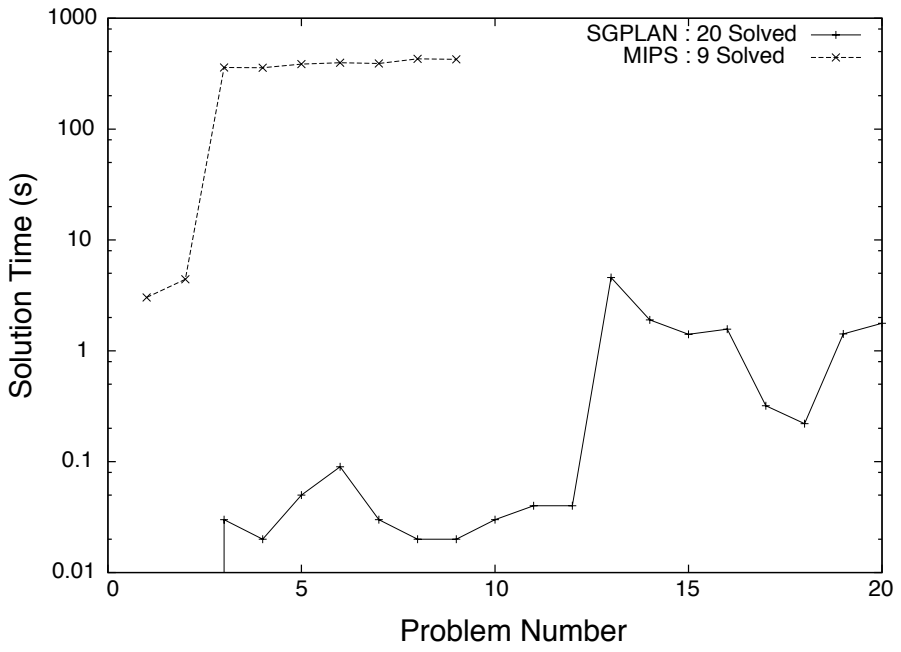
Conclusions on Satisficing Planning

Performance on purely temporal domains was very good, and temporal planning as a whole seems a very promising method. With suitable adaptations it could be leveraged to tackle Unit Commitment. The Airport domain was of particular interest as it was at a scale comparable to real-world situations. Undoubtedly modern versions of SGPLAN ran on modern computers could tackle the full MUC problems to a high standard.

There are many reasons to believe that the lack of guaranteed optimality from a satisficing planner does not imply a large gap to optimality. The fact that the satisficing planner LPG could solve temporal problems to the same level as the optimal planner CPT in the Airport domain is encouraging. Furthermore, the closeness of results in that temporal domain and the satellite domains when an optimal solution was unknown suggest a convergence towards the optimal solution. The fact that satisficing planners found optimal plans in domains where optimal planners were successful gives some intuitive (though unquantifiable) confidence that in domains where optimal planners are not successful or do not compete, when planners with very different search algorithms and heuristics produce very similar solutions, these act as an informative upper bound on the optimal solution. In cases where results



(a)



(b)

Figure 4.6: Results for the Temporal-Metric variant of the Openstacks domain from IPC 5. Results show that clearly SGPLAN has an advantage but if it is not performing the correct optimisation (as discussed in the main body text) then this performance could be misleading, implying planning performance on temporal-metric domains lags behind purely temporal domains.

vary quite dramatically no such intuitions can be claimed.

It should be noted that low problem coverage would not be acceptable for planners tackling Unit Commitment. It is essential that every problem be solvable to a point, even Branch and Cut algorithms stall on some awkward problem instances (see Appendix A.1.1). Thus if satisficing planners can demonstrate strong performance on a subset of instances, and fast solution times on those that show weaker performance, this is still a valid contribution and the features of those problem instances which posed a problem can be analysed further.

Finally, the ability to have variants of the same planner (HSP variants, LPG 3, 4) tuned and consistently performing for speed and coverage is encouraging for the flexibility of the system.

It is harder to assess the performance of satisficing planners on temporal-numeric domains with numeric optimisation. There appears to have been much less community-wide interest in these domains despite encouraging signs (see below for more details). Given the complexities of adapting the existing technologies to model these variants, it is not surprising that there were less participants able to even attempt those problems. The few entrants able to tackle them showed promise and improvements (eg. LPG 3 to 4, SGPLAN iterations). In the case of CPT in later competitions, the domain changes meant these planners were compared against others on a slightly uneven playing field. In some cases this resulted in unjustified demotion of ability, which perhaps slowed development of those promising systems. Whatever the reason, this area has not been exploited quite so much as others.

Whilst satisficing planners have shown potential in temporal-numeric domains this area has not been explored as deeply as purely temporal domains. This presents an opportunity for a strong temporal planner already adapted to tackle numerically constrained temporal optimisation, to be supplemented with additional features to strengthen its numeric optimisation performance. Through this work, initial thoughts on how this could be done for Unit Commitment are presented.

4.2.4 Conclusions on Planning for Optimisation

From this overview of benchmark problems relating to planning for optimisation it appears there have been great improvements in in problem coverage and solution time in non-temporal non-numeric domains for optimal planners. However, the lack of realistic problem coverage and relative performance on temporal-numeric domains compared to the best satisficing planners means they are currently not viable for domains like Unit Commitment. The apparent lack of optimal planners since TP4 and HSP^*_a which support temporal-numeric domains indicates that this is a less active area of planning research than satisficing planning.

On purely temporal domains satisficing planners have been able to reproduce the results of optimal planners where problem coverage overlaps. Given that satisficing planners have much greater coverage than optimal planners one cannot assess their quality on those domains where other solution methods do not exist. However, it is intuitive and not unreasonable to think that that they provide a reasonably close approximation when many results from different planners agree.

There were also discussions in post-competition literature regarding the difficulty of producing domains inspired by real-world applications. The airport domain, praised for its real-world scaling, did not fully represent the true problem, with the temporal minimisation applying to a subtly different quantity to the real world goal. This highlights the difficulty of modelling real world problems with planning.

It was also noted that many of the numeric features of the PSR problem could not be modelled at time of publication. Work by Piacentini et al [9] demonstrates modern approaches to tackling complex numeric problems, by linking to a specialised external solver from within a temporal planning framework. The success of this work demonstrates the feasibility of a similar method for Unit Commitment, proposed in Chapter 6.

Just as the introduction of Timed Initial Literals allowed the expansion of the Satellite domain to become more realistic, the introduction of Timed Initial Fluents should also allow, and indeed encourage, the formulation of domains such as Unit Commitment. Complex temporal-numeric domains such as this could become future benchmark problems.

Whilst planners can tackle certain problem instances to optimality, there has been less focus on driving the field in the direction of numeric optimisation in a temporal domain. Interesting problems such as Rovers (2002, 2006) and Satellite (2002, 2004) feature simple numeric goals but place a large focus on solution quality relating to numeric quantities. These, nor similar domains, have appeared in later competitions.

The domain variants for Openstacks (2006, 2008, 2011) also reveal shift in focus with the minimisation metric changing from a combination of minimising the number of stacks and total time to minimising make-span. Also telling is the reuse of certain domains over others. Chemical Pathways and TPP were both interesting real-world domains in which only one planner successfully competed. Inclusion in later competitions to test the optimisation qualities of new SGPLAN and MIPS iterations would be welcomed and promote research into numeric optimisation in a temporal domain.

As the competition progressed through the years its scope naturally expanded but has not gone deeper into numeric optimisation. It is clear from collating this review how the number of domains with numeric optimisation variants has dropped with each competition iteration.

The diminishing interest in those kinds of domains in later competitions is disappointing. As noted above, for many real-world problems that are difficult to model using other techniques[20] the inclusion of features such as probabilistic planning, soft goals and preferences, and incorporating machine learning help to widen the scope of problems planning can handle. This allows it to be one of few solution methods able to tackle such domains.

Whilst there is not an abundant collection of temporal-numeric domains with numeric optimisation as benchmarks in the IPCs, there are enough to pique interest. The research above suggests that with adaptations, planners could successfully tackle

[20]Domains which would be difficult to model using other methods including the Airport domain, planning the complex interactions of simultaneous printer jobs [186], planning the manufacturing process of LCD displays [187], and a Home-Monitor system to interact with vulnerable people helping them complete of day to day tasks [140].

Unit Commitment.

Unit Commitment is a problem with known optimal solutions and an easily satisfiable goal, if one does not consider optimisation. The survey above indicates some precedence that these kinds of problems are both non-trivial and of interest to the planning community.

The inclusion in the IPCs of known hard domains such as Openstacks and TPP, the real world applications of Satellite, Rovers, Airport and Chemical Pathways, and the modelling of complex domains such as PSR which could not be fully modelled at time-of-publication, shows the community's desire for planning to tackle real world problems, not just conceptual problems, solely of academic interest.

A widening of interests, continued inclusion of new features coupled with retained support and much active research in original and previous streams in the IPCs shows that the planning community is growing in active participants and research output.

The fallibilities of the optimal planners CPT and Semsyn not handling the full complexity PDDL, and the refocussing of the IPC 7 competition on temporal domains with required concurrency, show that this field is still relatively new and developing.

Both this ambition and increased widening interest are promising signs that the Unit Commitment problem will be received well by the community and that further work in this area would likely follow. With this in mind the following section details the specific features of Unit Commitment that make it an interesting challenge for Automated Planning.

4.3 Novelty of Unit Commitment to the Planning Community

It appears from this review that whilst numerical optimisation within a temporal domain has seen less attention than other aspects of planning, the potential to tackle these problems has been demonstrated. As many real world problems demonstrate complex temporal structure with a goal of numeric quality this is an area it would be beneficial for planning to expand into. Unit Commitment is an excellent candidate for demonstrating and testing this potential.

Forming this model was non-trivial and full details are given in Chapters 5 and 6. Two very different approaches are presented, one being solvable by any planner handling the required constructs, and one which requires a dedicated solver. There is thus a debate to be had over the merits of both approaches.

Much emphasis is placed on the general applicability of planning algorithms in the literature, creating black-box solution methods. These would favour the approach demonstrated in Chapter 5 where the problem could be handed to any planner. This has the advantage of the community developing planners with a high problem coverage over a much wider class of domains. Thus research not dedicated to Unit Commitment can bring performance gains to the problem.

There is also an advantage in developing a domain specific system. Domain specific adaptations to Meta-Heuristic methods (see Chapter 3) showed much improvement in solution quality over more generic approaches. Whilst these systems may have coverage in a reduced class of domains, the solution quality in the domains tackled is likely to be much higher.

Perhaps a focus on general black box applicability has restricted the breadth of problems planning presented and tackled as part of the IPC, and thus restricted development of specialised planners which have not received the same exposed gained from participation in the IPC.

Chapter 6 details a domain specific approach for Unit Commitment with planning. Whilst this is only applicable for this problem the general techniques could be abstracted out after further research. This would open up a whole class of problems focussed on numerical optimisation that could be tackled in a similar manner, as was the case in [188] discussed in more detail below. Successful implementations of domain specifics planner can help to expand the scope of planning as a whole, which is clearly of benefit to the community.

This problem also represents a real-world example of interesting language constructs which have received less attention than others. Exogenous processes are critical for modelling many problems, especially in engineering. These represent the impact of systems the planner is not reasoning about on the system it is reasoning about.

In many examples, especially examples from power systems engineering (such as the PSR domain above, and work such as [9, 14, 15]), these processes bring about a numeric effect. Modelling these is a simple extension of Timed Initial Literals which have been included in ICP domains since 2004 and have received much attention. There is work in the literature that handles TIFs (see for example [188]) however the new problems throw up new challenges, previously not considered. POPF, a very established planner, had bugs in its handling of TIFs discovered during development of the first planning model. This demonstrates that existing language constructs used in new settings can still throw up interesting challenges and provide opportunities for the development of novel planning approaches.

Furthermore the planning community has shown an interest in real world problems such as Unit Commitment, and has shown success at handling problems within the field of power engineering. Work by Bell et al [14, 15] presented a successful implementation of planning to reduce the cost of a system widely used to manage the tasks of a power substation. The planning approach produced plans of a lower cost and with fewer actions, reducing wear and tear of equipment, than the existing system.

Work mentioned above by Fox, Long and Magazzeni [188], presented a problem similar to Unit Commitment. There a single demand was to be served by a collection of batteries over a given horizon. Unit Commitment includes some key complexities over the battery problem which would represent a considerable advance for planning technologies if successfully solved.

Firstly, Unit Commitment requires the concurrent scheduling of units to serve the demand. As discussed above, correct concurrent reasoning is complex. In the case of Unit Commitment it greatly increase the combinatorial element of serving the demand at any instance. This increases the branching possibilities at each decision point placing more emphasis on having an efficient and effective heuristic to ensure the branches are evaluated quickly and the only the most promising ones expanded.

Secondly, the goal in Unit Commitment is more complex than in the battery problem. The cost in Unit Commitment is the metric, the duration of the plan is fixed to the given horizon, whilst in the battery problem the goal is to maximise

the total time taken. This is a goal that has featured in many benchmark planning problems and was very effectively tackled in this case. The costs involved in Unit Commitment which contribute to the metric are themselves very complex to model, discussed further in Chapter 5.2.4.

The Battery problem also represents a case where a domain specific heuristic can be generalised. The authors cast their problem into a general class featuring a *"monotonically decreasing resource"* where *"the longest plan is required"* [188, p78]. This advocates the developments of domain specific solvers, which after theoretical work classifying the problems tackled, can being generalised to a wider set of problems.

The scale of Unit Commitment is much larger. [188] considers problems of up to 8 batteries whereas for Unit Commitment the test systems proposed start at 6 units, and small scale realistic implementations would be for around 50 units, and large scale industrial implementations are for hundreds of units. Increasing from 8 to tens or close to a hundred units is likely to introduce many complications which cannot be predicted a priori.

Finally Unit Commitment represents a problem with a goal featured but not fully explored in the planning literature. The planning horizon is essentially fixed and the goal focusses entirely on plan quality based on a numeric metric. The was very similar to the Satellite problem mentioned above, and alters the temporal reasoning required from the planner. As discussed in Chapter 5.4 the relaxed planning graph framework does not appear effective in providing guidance for this highly numeric problem, and new options are proposed.

To conclude, Unit Commitment represents a real-world temporal-numeric domain which has not previously been explored in planning. It has many interesting features including required concurrency, complex relationships between goal constraints and TIFs, and a fixed planning horizon with a strong focus on numeric plan quality. Existing methods provide bounds to optimality which facilitates an accurate assessment of the performance of a planning approach. Therefore this problem represents an excellent opportunity to demonstrate the flexibility and potential of planning. It also has the potential to extend the scope of planning in a different

direction than is currently the focus of the community.

4.3.1 Choice of Planners

Timed Initial Fluents are not specifiable in any competition version of PDDL (see the 2011 IPC specification [189] and note that `<init-el>` has no equivalent to the Timed Initial Literals construct `(at <number> <literal(name)>)` for Timed Initial Fluents). Furthermore there are few planners with documented supported for timed initial fluents. Overall, the choice of planners to develop is restrictive.

The satisficing planner, POPF was chosen to be extended, as Piacentini et al [9] were developing a version to support Timed Initial Fluents alongside an exogenous heuristic, necessary for the modelling of Unit Commitment.

As background, POPF2 competed in IPC 7 and came an overall 4th in the temporal satisficing domains[21] and so was a good base from which to think about developing a system to handle Unit Commitment. Piacentini et. al [9] developed a system with similar planning requirements to those necessary for Unit Commitment at the same time as this work.

Presented at the International Conference on Automated Planning and Scheduling (ICAPS) in 2013, the work received an award for the best student paper in the novel applications stream, further demonstrating the community's interest in real-world problems such as the one presented therein, and Unit Commitment.

[21] Table 4.5 shows the 'score' for POPF2 in each domain, which position it was in and the score of the best planner taken from the results available in [190]. The totals for each planner were summed to form an overall ranking. More details on how the results were calculated can be found at [190].

Criteria	Ranking	Score vs Best	Planner
Number Solved	4th	119 vs 145	yahsp2-mt
Solution Time	3rd	89.31 vs 136.23	yahsp2-mt
Quality	3rd	110.45 vs 126.5	daeyahsp
Time + Quality	4th	114.12 vs 138.45	yahsp2-mt
Overall	4th	432.88 vs 531.05	yahsp2-mt

Table 4.5: Table showing the comparative scores from IPC 7 of POPF2 against other entrants in the temporal-satisficing domains. The entrants above POPF2 were satisficing evolutions of the HSP planner family.

Chapter 5

Initial Planning Model

5.1 Motivation

Chapter 4.3 presented a discussion of why studying Unit Commitment as a Planning problem would represent a contribution to the planning community. Below, some key reasons why a planning approach for Unit Commitment would be beneficial from a Power Systems Engineering perspective are presented.

- The key part of Unit Commitment is the temporal online / offline schedules produced, not the exact unit outputs, and planning has very strong temporal reasoning as demonstrated through Chaper 4.2.
- The main feature of a MIP formulation for Unit Commitment that makes the problem hard to solve is the number of binary variables with complex constraints covering them. These all come from the temporal sub-problem, implying a weakness of the MIP is a strength of planning.
- Unlike most Meta-Heuristics approaches presented in the literature (discussed in Chapter 3), a planning approach uses an entirely different formulation of the problem. Thus it does not inherit the discretisation problem seen in a MIP model, which other meta-heuristics do.
- Analysing a MIP solution to form schedules and thinking about how these would be applied in practice reveals that there are clearly fewer times when

a unit's online status switches, an event requiring an action in the power system, in the final schedule than binary decision variables in the MIP. The action formulation of a planning implementation is therefore a more natural fit to this problem than a large collection of binary variables forced over a discretisation of time.

- Practical approaches for Stochastic Unit Commitment are still in development. There are many interesting implementations of planning handling uncertainty, suggesting that a more sophisticated implementation than just performing multiple runs is a possibility. Before such complex planning systems can be developed, an efficient formulation of deterministic Unit Commitment is required.

Before detailing proposed planning models, these points are expanded for clarity. First consider the temporal side of the problem. Suppose one was to manually solve the problem of finding online / offline schedules for a unit. An intuitive approach would be to decide on the times one would switch on a unit, and how long it was on for, i.e. when one would switch a unit off. Approaching the problem asking "at 12:00 midnight will it be on or off? At 1:00am will it be on or off? At 2:00 am ... etc." seems less natural.

Supposing no knowledge of research in either planning or MIP formulations one would attempt to solve the first of these formulations, which adheres closely to the structure of the planning problem. Planning as a discipline is much newer, and has consequently received much less attention, than MIPs and Branch and Bound. Simply because it has not yet had a chance to demonstrate the breadth of ability demonstrated by modern MIP solvers, backed as the are by multi-million dollar companies, should not dismiss it from consideration a priori when the formulation would appear more natural.

The solutions produced and overall solution methodology are also more succinct than a MIP. Consider a peaking unit with low start up cost but high marginal cost, minimum online time of 2 hours and minimum offline time of 1 hour. This type of unit will typically be used only during the morning ramp and evening peak. As detailed in Appendix A.1 a MIP model would have variables $o(u, i)$ for $i =$

i	1	2	3	4	5	6	7	8	9	10	11	12
$\mathrm{o}(u,i)$	0	0	0	0	0	0	**1**	**1**	1	1	**0**	0
...	13	14	15	16	17	18	19	20	21	20	23	24
$\mathrm{o}(u,i)$	0	0	0	0	**1**	**1**	1	1	**0**	0	0	0

(a)

360	switch_on(u)	[**120**]
600	switch_off(u)	[**60**]
960	switch_on(u)	[**120**]
1200	switch_off(u)	[**60**]

(b)

Table 5.1: Possible schedule for a peaking unit (one which can be brought online quickly, typically only remaining online for a few hours at a time, used primarily to serve periods of peak load) with minimum online time of 2 hours and minimum offline time of 1 hour, used during the major morning and evening ramps in demand. (a) would be a MIP representation and (b) a planning representation. The periods representing minimum online and online times are in bold type. They highlight how a Planning expression of a Unit Commitment schedule is more natural than a MIP expression.

$1 \dots m$ denoting the time period and u indexing the unit. A planning model would determine timestamped actions.

Table 5.1 gives the same solution for this unit over a 24 hour horizon. Supposing the search in the planner can be correctly guided and implemented efficiently it only has to reason about 4 decision points with regards to this unit. Branch and Bound has to solve for 24 binary variables. Whilst the cuts and constraints detailed in Chapter 2.3 tightly group some of the variables to ensure minimum online and offline times are respected, all others must be reasoned about. In the case of this peaking unit, more variables are not covered by minimum online / offline constraints than those that are.

The planner however will append a `switch_on` action to the plan, whose duration is forced to the minimum online period (see below for further details), so this constraint is enforced without the need for additional constructs. The same occurs for the `switch_off` action. A further and perhaps more rewarding benefit is that once these actions have been applied, the online status of the unit remains in a steady state. The planner does not have to re-evaluate online status at fixed time intervals, which the Branch and Bound solver would have to consider.

The object-action paradigm of planning and resultant solution appear a more natural fit for Unit Commitment as well as being more intuitive to understand, and motivates the development of this initial method. A first attempt at modelling Unit Commitment as a planning problem is presented in the remainder of this chapter. The key features are discussed using the PDDL for the domain. Full code can be found in Appendix B.2.

5.2 The Model

5.2.1 Temporal Constraints

A planning approach was in part motivated due to the simple and rigorous way in which the temporal constraints on generating units can be modelled. Two simple actions replace the complex constraints in a MIP that some meta-heuristics have difficulty enforcing. There, minimum online and offline constraints are often relaxed

during state generation as the random elements of the search will most likely generate states which violate them. This causes extra reasoning and slows the search down (further details are given in Chapter 3). In a Branch and Bound setting, much research was needed to efficiently tighten the minimum online and offline constraints (as discussed in Chapter 2.3). As discussed below, a Planning formulation would not suffer from the same complications surrounding these critical constraints.

The `switch_on` and `switch_off` actions are set to be the length of the minimum online or offline period respectively and a system of predicates is used to ensure that one of these actions cannot be interrupted by another. This simple formulation is all that is required to enforce these constraints. Mechanisms to improve solution speed, such as the cuts introduced by Rajan and Takriti in [41] or domain specific mutation operators as in Meta-Heuristics (see [93] for example), are not necessary.

Predicates `on` and `off`, representing the generating unit's online status, are switched with the actions `switch_on` and `switch_off`, shown in Figure 5.1. Clearly before a unit can come online through `switch_on` it must be offline. The second precondition `canSwitchOn` is a flag which is negated when a unit comes on and only becomes true again with the effect `(at end (canSwitchOn ?u))` of the `switch_off` action. Similarly `switch_off` has a precondition `canSwitchOff` only made true by the effect `(at end (canSwitchOff ?u))` of `switch_on`. In this way `switch_off` must follow the end of `switch_on`. By fixing durations of `switch_on` and `switch_off` to the minimum online and offline durations respectively each unit is forced to be online / offline for at least its respective minimum online / offline period. This is illustrated in Figure 5.2.

The benefit of this action choice is that the constraints are enforced but the actual online and offline times of a unit are not fixed. There are no upper bounds on the time a unit can be left online or offline for as the actions don't negate the precondition they switch. The actions themselves however do have a fixed duration, not expressed as an inequality, making for simpler more efficient reasoning by the planner.

```
(:durative-action switch_on
 :parameters (?u - unit)
 :duration (= ?duration (minimumOnTime ?u))
 :condition ( and
    (at start (off ?u))
    (at start (canSwitchOn ?u)))
 :effect ( and
    (at start (on ?u))
    (at start (not (off ?u)))
    (at start (not (canSwitchOn ?u)))
    (at end (canSwitchOff ?u))
    (at start (assign (output ?u) (generationMin ?u)))
    (at start (increase totalCost (costStartUp ?u)))
    ))

(:durative-action switch_off
 :parameters (?u - unit)
 :duration (= ?duration (totalOffTime ?u))
 :condition ( and
    (at start (on ?u))
    (at start (canSwitchOff ?u)))
 :effect ( and
    (at start (not (on ?u)))
    (at start (off ?u))
    (at start (not (canSwitchOff ?u)))
    (at start (assign (output ?u) (generationMin ?u)))
    (at end (canSwitchOn ?u))))
```

Figure 5.1: The two actions responsible for switching the online status of a unit and enforcing the temporal constraints.

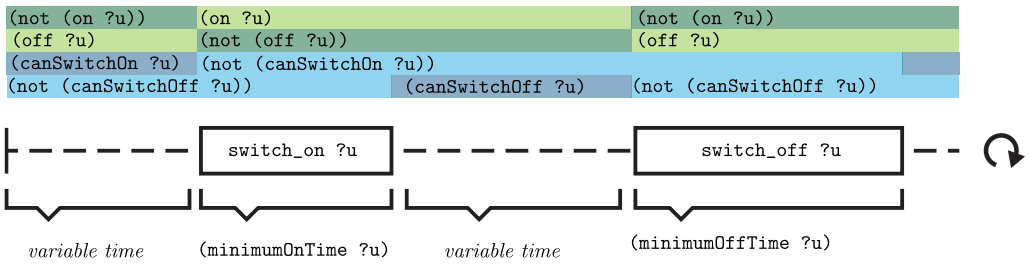


Figure 5.2: The two actions switch_on and switch_off can be scheduled to give any length online and offline periods despite having fixed duration. The temporal constraints are enforced using the predicates canSwitchOn and canSwitchOff.

5.2.2 Modelling Supply and Demand

This initial model includes the exact output for each generating unit in the system. A fluent `output` is associated with each unit and a total `supply` fluent are used to model this and the overall supply.

To have variable changes in output requires an action with a variable duration. Two actions, `ramp_up` and `ramp_down` (Figure 5.3) alter the `output` and `supply` fluents at a fixed rate per minute with variable duration. This allows the planner to set the output of a unit to any level thus performing the Economic Dispatch sub-problem.

The formulation of these actions is reasonably standard. Note however the necessary inclusion of the `#t` implying a continuous process. It is not possible to specify an action which sets the value of a fluent to a value to be determine at run-time. Therefore it is not possible have an action with no duration that instantaneously ramps the unit to a value between its minimum and maximum, as occurs in a MIP solution.

These actions are complex to reason about as they require continuous reasoning about the `#t` parameter. This action construct therefore requires complex and time consuming reason, and is an influential factor motivating the development of the second model presented in Chapter 6.

```
(:durative-action ramp_up
 :parameters (?u - unit)
 :duration (<= ?duration (/ (- (generationMax ?u)
                  (output ?u)) (rampRateUp ?u)))
 :condition ( and
    (over all (on ?u))
    (over all (not (rampingDown ?u)))
    (over all (<= (output ?u) (generationMax ?u))))
 :effect ( and
    (at start (rampingUp ?u))
    (at start (increase (supply) (output ?u)))
    (at start (increase (totalCost) (+ (* ?duration (output ?u))
            (* (* 0.5 ?duration) (- (+ (output ?u)
                (* (rampRateUp ?u) ?duration)) (output ?u))))))
    (increase (output ?u) (* #t (rampRateUp ?u)))
    (increase (supply) (* #t (rampRateUp ?u)))
    (at end (not (rampingUp ?u)))
    (at end (decrease (supply) (output ?u)))))

(:durative-action ramp_down
 :parameters (?u - unit)
 :duration (<= ?duration (/ (- (output ?u) (generationMin ?u))
                     (rampRateDown ?u)))
 :condition ( and
    (over all (on ?u))
    (over all (not (rampingUp ?u)))
    (over all (>= (output ?u) (generationMin ?u))))
:effect ( and
    (at start (rampingDown ?u))
    (at start (increase (supply) (output ?u)))
    (decrease (output ?u) (* #t (rampRateDown ?u)))
    (decrease (supply) (* #t (rampRateDown ?u)))
    (at end (not (rampingDown ?u)))
    (at end (decrease (supply) (output ?u))))))
```

Figure 5.3: Actions representing a unit's ramping. These alter output and supply and a fixed rate per minute but with variable duration allowing the planner to perform Economic Dispatch.

```
(:durative-action envelope
 :parameters ()
 :duration (= ?duration 1440)
 :condition (and
         (at start (> supply demand))
         (over all (> supply demand))
         (at end (> supply demand))
         (at end (complete))
         (at start (preconditionToStart)))
 :effect (and
         (at end (demandServiced))))
```

Figure 5.4: Action to enforce the system balancing constraint. Predicates are used to ensure this action is scheduled at the same time as Timed Initial Fluents updating the demand.

5.2.3 Balancing Supply and Demand

Customer Demand is represented by a fluent `demand` and updated using a series of Timed Initial Fluents given in the problem file. Balancing supply and demand is enforced by the temporal action `envelope`, Figure 5.4. `envelope` has a fixed duration equal to the length of the planning horizon and requires the fluent `supply` discussed above, to be greater than the fluent `demand` throughout its duration.

The action contains three more predicates which combine to ensure this action is run for the first x minutes of the planning horizon, assumed to be the same period as the Timed Initial Fluents updating `demand`. The goal requires the two predicates `demandServiced` and `complete` to be true. `demandServiced` is initially false and only set to be true by `envelope`, forcing the planner to run this action. `complete` is initially false and set true using a Timed Initial Literal at the end of the planning horizon. The condition `(at end (complete))` of `envelope` ensures this action is not run too early. In the problem file `(precondition-to-start)` and `(at 0.005 (not (precondition-to-start)))` coupled with no action set-

ting `precondition-to-start` to true means the planner must schedule the action `envelope` in [0, 0.005). As this action must run and has a duration the entire length of the planning horizon, the constraint `(over all (> supply demand))` is always enforced.

5.2.4 Modelling Cost

Finally the cost is to be modelled. The No Load cost is linear in the duration of the action. This can be seen in Appendix B.2 and is modelled as a simple increase start effect in the actions `ramp_up`, `ramp_down`. As the action must have ended before the plan will be considered complete, it makes no difference whether the cost is incremented at the start or end of the action.

These two actions alone do not account for the times when a generating unit's output remains static. In order to ensure the marginal and no load costs can be incremented during these times there must also be a third action.

The `generate` action can be seen in Appendix B.2, and increases the total cost by the no load cost incurred during that action's duration. There is however no intrinsic reason for the planner to schedule this action, as all it does is increase the cost, negatively impacting the metric in the problem goal.

This necessitates the deduction of a unit's contribution to the overall supply at the end of a ramp or generate action. By removing the output of a unit from the total supply at the instantaneous end of an action necessitates the immediate addition of another action. All three ramp and generate actions increment the total supply with the unit's output so if there are no breaks in a schedule between these units then the supply will remain above demand and the problem solvable, forcing the planner to schedule these actions. This is a workaround which complicates the planners reasoning, and is a key factor to the development of the second planning model.

The marginal cost is more complex. As the ramps are being modelled as a continuous increase, it would be ideal to model the costly similarly. The amount by which to increase output and supply when ramping up is linear:

$$g(u,t) = g(u,t_0) + t \cdot R_+(u) \tag{5.2.1}$$

where g is the output and t_0 is the time the action is started, t is the variable of concern as the duration is the variable the planner is resolving to find.

The updates for the cost however are nonlinear. The marginal cost of ramping up from g_0 to g_1 in time d is given by:

$$C_{\text{ramping}}(u) = C_{\text{marginal}}(u) \cdot \left[d\ g_0 + \frac{1}{2}(g_1 - g_0)d \right] \tag{5.2.2}$$

g_0 is the output at the start of the action and g_1, the final output, is dependent on the duration. Expanding this expression shows this cost is quadratic in the duration as well as containing a product of two variables, duration and g_0.

$$C_{\text{ramping}}(u) = C_{\text{marginal}}(u) \cdot \left[d\ g_0 + \frac{1}{2}((g_0 + R_+(u)d) - g_0)d \right] \tag{5.2.3}$$

$$= C_{\text{marginal}}(u) \cdot \left[d\ g_0 + \frac{1}{2}R_+(u)d^2 \right] \tag{5.2.4}$$

The same equations are easily calculable for ramping down. Only linear expressions can be handled in Popf at time of development, meaning the marginal cost of ramping cannot be accurately calculated. This was another contributing factor in pursuing the second model presented in Chapter 6, discussed in more detail in Section 5.4.

5.3 Soving Unit Commitment using UPMurphi

The only planner available at time of development able to handle all of the features required in the above Unit Commitment model without modification to the planning code is UPMurphi.

The initial goal was to use the Popf planner, which has better concurrent and continuous reasoning than UPMurphi. At time of development Popf could not handle the Timed Initial Fluents interacting with the overall constraint of supply being greater than demand. Since this initial development, work by Piacentini et al [9] has enabled Popf to handle this aspect however there remain open problems. The non-linear marginal cost during ramping remains a problem.[1] The decision was made to run initial tests using UPMurphi.

[1]The model as presented here would also crash Popf as the negative over all conditions in the

UPMurphi is a planner which implements a 'discretise and validate' approach. This means that time is dynamically discretised, an action applied and the resulting state checked for validity.

The search implemented is an A* search based on the heuristic value in each state. The state with the highest heuristic value is the next state expanded, to which all actions are applied. The resulting states are added to the explored tree of states and the state with the highest heuristic is again advanced.

Unlike a traditional planner where a predefined heuristic function is applied to each state, having the heuristic value as a variable in the state, free to be modified by any action, gives greater flexibility with the heuristic. See section 5.3.1.1 for a discussion of the most successful heuristic and how it was applied to a subset of problems.

As UPMurphi handles time using the discretise and validate approach, durative actions such as ramping are split into start and end actions. Consequently as states are resolved in the 'check' stage of applying an action, the non-linear costs can be incremented as part of an instantaneous start or end action.

This leads to the set of actions being as in Figure 5.5a and predicates and fluents being as in Figure 5.5b and Figure 5.5c. Of note over the Pddl model presented above is the "Pass Time" Action, specific to a UPMurphi implementation, which advances the time state variable and performs the validation. The validation checks none of the hard operating constraints are violated, and if they are, removes the state from search tree.

5.3.1 Model Development With UPMurphi

This section details the mode development process and qualitative results of running the above planning model on UPMurphi. Domain formulation always has an impact on problem performance and so altering the action structure and manipulating the UPMurphi search process was done hand in hand in an attempt to find the

ramping and generate durative actions are not handled correctly. This can be avoided by using extra predicates however have been omitted from the model description here for clarity.

Action
Pass Time
Start Ramp Up
Stop Ramp Up
Start Ramp Down
Stop Ramp Down
Switch On
Switch Off

(a) Actions

Predicates
Online
Ramping Up
Ramping Down

(b) Predicates

Fluents
Output
Timers
...

(c) Fluents

Figure 5.5: A list of boolean predicates, numeric variables (fluents) and actions used in the UPMurphi AI Planning model. Note that the *Timers* represents a suite of numeric variables which track how long a unit has been on / off for, ramping for, when each of those actions occur etc. to allow for tight preconditions as discussed in 5.3.1.3.

best overall planning system. The problems used for testing and development of this model were all variants of the systems given in Appendix A.1.1, using 4-12 generating units in various test portfolios with the demand data outlined in Appendix A.1.1.

5.3.1.1 UPMurphi Heuristic

The heuristic function at a node n in an A* search is typically the cost to reach the node n plus the expected remaining cost to reach the goal. This did not provide an informed search as there is no quick and (moderately) accurate way to estimate the remaining cost to go, and the total cost of plans towards the end of the planning period were so much greater than those at the start that the planner simply enumerated all of the early plans before moving forward in time.

A weighting of the total cost so far given the time of the current state did not appear to improve the search even when only one unit was necessary to serve the load. This is potentially down to the cost being in the order of 10^6 and time being

in [0,1440] so the cost overwhelms the time weighting.[2]

To study the effects of changing the heuristic, Economic Dispatch problems were studied (problems where the units initially on could serve the entire load profile). A 'greedy' heuristic was developed based on the current cost per minute of a plan given the current load being served weighted with the proportion of the planning horizon remaining (see Appendix B.1 for details).

This heuristic was tested, and coupled with the preconditions discussed in section 5.3.1.3, proved effective for problems of up to 10 units where none of those units needed switching off or further units switching on. The unit loading matched (bar symmetric units) the outputs from the MIP model and solution times were comparable. Whilst this is positive for the development of the planning model it is of course redundant as a standard LP (not MILP) could easily solve large deterministic Economic Dispatch problems. The difficulties of adapting this heuristic to dispatch problems are discussed in section 5.3.1.4.

5.3.1.2 Dynamically Discretising Time : The "min_timer" Function

"Pass Time" advances time to the next point an action may need to be applied. One downside of UPMurphi's dynamic discretisation experienced throughout testing was that if the discretisation was too narrow too often, the state space grew exponentially and the search essentially stalled with many states all appearing identical to the planner. As theoretically an action may be applied at any point in time one could potentially advance a state by anywhere from 1 min to 30 minutes (a typical gap between time points in Unit Commitment MIP Models) however this creates a huge amount of near identical states.

To overcome this a `min_timer` function was developed to calculate when the nearest point in time a corrective action could be applied is. This value was the amount to advance by, and if informative, allows for all 'sensible' actions to be tested and fairly evaluated without generating a vast intractable state space. Explicitly, for the Economic Dispatch problems studied to develop the heuristic, the minimum

[2]Dramatic scaling could be implemented but a scheme that worked well in multiple problem instances was not found.

time the planner could advance by was taken as the minimum of:

- the time until the load changes
- time until the system globally ramps below the current demand
- time until the system globally ramps beyond the next demand
- times restricted by individual ramping units
- the minimum time the system can wait until some ramping could have to occur to reach the next demand level

This `min_timer` function, illustrated in Figure 5.6 was thought to be a key area of development if this implementation were to succeed. If it over estimates the time until the next action point, avoiding action to critical situations (such as ramping beyond limits or ramping unnecessarily high) may not be taken and states will be incorrectly missed. If it under-estimates the time many excess states are explored increasing both memory usage and solution time.

In the small Economic Dispatch problems used to aid development of the heuristic, this proved effective reducing the solution times on the small test systems used. What was found was that when large models were used to stress the system, too many states were still being generated despite this more domain specific way of performing the dynamic discretisation. The resulting poor scaling to real scale problems is prohibitive for Unit Commitment where typical systems will require up to a hundred units or more.

5.3.1.3 Pruning Through Pre-Conditions

Effective use of an actions's preconditions to prevent the planner from creating states which will necessarily be either duplicates or lead to dead ends should reduce the explored portion of the state space reducing solution time.

The start and stop ramping actions were thus modified to make use of knowledge of the domain, preventing the planner from creating states guaranteed to be dead ends once time is passed. The conditions are:

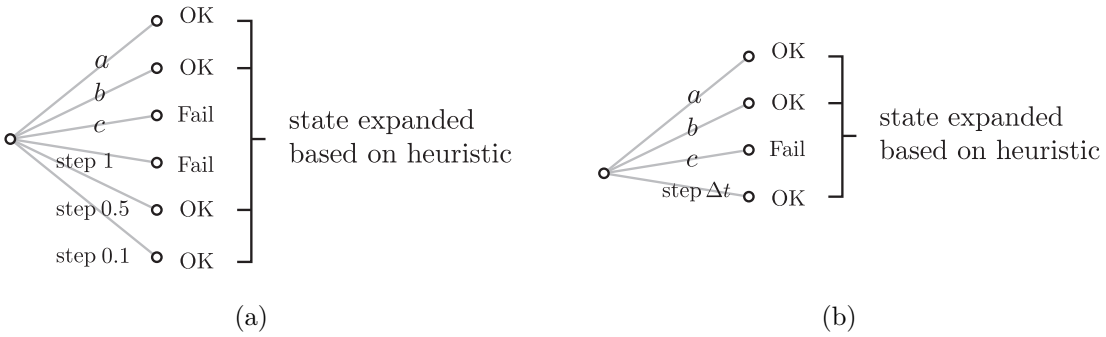


Figure 5.6: Schematic illustrating how the standard UPMurphi approach and min_timer approach differ. In the standard approach UPMurphi will apply all actions to a state, including advancing time by a series of decreasing intervals, checking each state for validity, as in (a). In this way, if an action is applied and then time advanced by a large amount breaking a constraint, the finer granularities will arrive at a state where the action has still been applied but only advanced slightly further down the time line, so necessary actions to resolve what would break the constraint can be taken. In the Unit Commitment problem the number of time steps to attempt would be very large as many combinations ramping actions could be used between each change in demand, and switching actions will have to be tested too. This results in a huge state space and is not efficient. Instead time is only advanced by one amount, dynamically calculated in the min_timer function, as in (b). This reduces the state space over the default approach.

- If the current output is such that ramping for 1 minute[3] would cause it to be outside of its operating limits, then ramping cannot occur.
- If supply is such that ramping down for one minute would mean demand is no longer below supply then ramping down cannot begin.
- Starting / stopping ramping for the same unit cannot directly follow stopping / starting ramping unless some time has elapsed to prevent states being generated which could occur from leaving a unit ramping.
- If current supply is never exceeded by future demand, global ramping up (combined total of ramp rates being positive) cannot occur as this will never be optimal.

It was found that effective use of preconditions preventing duplicate states improved solution time greatly. If one were to continue developing this model to dispatch units (rather than advancing to the separated model of Chapter 6), restricting when units could be switched on or off (based on more than just the minimum on / off times) would be crucial to attaining competitive performance against the MIP.

5.3.1.4 Dispatching Units

The heuristic detailed above is a greedy heuristic but provided an efficient and effective heuristic for Economic Dispatch problems. Dispatching units requires some notion of look-ahead, i.e. it is not just the instantaneous start up cost that will affect which unit should be switched on, but how costly it will be for the unit to serve the extra load over much longer periods of time.

Accounting for this in the heuristic proved challenging. When the start up cost was included in the cost per minute calculation given in Appendix B.1, the heuristic value was dwarfed by this amount and search never returned to that state. Not

[3]As we are storing time as an int to reduce memory usage this "for 1 minute" is required as if the unit is not at its boundary but would reach it in less than a minute the planner may keep trying to ramp even though it is not possible.

including the start up cost in the heuristic meant the unit with the lowest marginal cost was always the one to be turned on.

The `min_timer` function discussed above was left unchanged when considering unit dispatch. As units could theoretically go online or offline at any point in the planning horizon, a `switch_on` or `switch_off` action would always be applicable. A simplifying assumption, that a unit could only go online or offline at the same moment as a demand change, was made to prevent state explosion during search. As the `min_timer` function already took the demand change time points into account, unit dispatch did not need to be accounted for there. It is possible that this model formulation choice impacted performance of the planner, however for the reasons discussed below (see Section 5.4), it was thought that there was more promise in developing a dedicated system (detailed in Chapter 6) than in pursuing this aspect further.

Another complication is to take into account how long the unit serving the surplus needs to be on for. If demand is only high for a short period of time a more expensive unit which can turn off quickly is preferable. Creating a heuristic with good performance retaining elements of the myopic case, which can also reason over the longer periods of time required when considering unit start up proved challenging and an effective solution was not found.

5.4 Conclusion from the UPMurphi Implementation

This model demonstrated that Unit Commitment can be modelled as a planning problem and be solved to an extent with 'off-the-shelf' planners. As discussed above performance gains could not be found, thus this model cannot be deemed a success.

One key reason could be that the formulation of the ramping and generate actions to model the Economic Dispatch sub-problem are both complex and a little convoluted. The output of a unit cannot be variably altered by an instantaneous action. The only way to model the actions so the planner can alter the output of a unit is by using a linear increase in a variable duration action. This is the formula-

tion used however requires the planner to support the `#t` language construct. This is complex to handle and could contribute to the long solution times.

Accurately modelling the costs involved also proved problematic. To track the marginal cost an action, during which the unit's output would be constant, had to be included in the model. The marginal cost could not be included in the `switch_on` action as the output was changing dependent on the other actions being concurrently scheduled. This meant a `generate` action had to be included, the purpose of which was to be scheduled between the ramping actions to increment the total cost correctly.

With either a ramping or generate action being scheduled for the entirety of the time the unit is online, the no-load cost could be incremented in each of those actions. This meant that the switching actions did not need variable durations and could be modelled as discussed in Section 5.2.1.

However it forced extra complexity into both the model and the reasoning process. The motivation for using a planning formulation was the natural action formulation reducing the number of decision points. By forcing these three actions to be directly sequentially scheduled for all times the unit is online increases the number of decision points and reduces the impact of the more efficient temporal reasoning.

Finally the cost of ramping could not be accurately modelled in POPF. A workaround is to assume the marginal cost throughout the ramp is the marginal cost as if it were running at the output at the start of the action throughout the actions duration. This is not ideal but implies similar instantaneous marginal cost increases inherent in the MIP model, where the output and cost changes instantaneously at point of demand change. Here the ramp down assumption is the same. The instantaneous ramp up is an over estimate compared to the MIP as ramping up occurs before the demand changes.

The less accurate cost model and current restrictions in POPF at time of development meant this model could not be tested on POPF. To begin investigating the possibility of implementing a planning system it was decided initial tests would be conducted using UPMURPHI.

Two key difficulties found when testing UPMURPHI likely to have been respon-

sible for the poor performance were implementing an effective look-ahead heuristic and the reliability and flexibility of the `min_timer` function.

The heuristic was either inherently myopic and therefore less informative, or inefficient to calculate. Given the complexity of the model and therefore how often the heuristic was being evaluated, it would not be beneficial to implement a more accurate heuristic.

The dynamic discretisation accelerated by `min_timer` worked well in the small test systems however heavily relied on the deterministic nature of the problem. As the move to stochasticity would be done using machine learning on deterministic cases this may not prove to be a problem. However this discretisation method may prove ineffective as the number of units grows and the points at which actions could occur increases.

The function could also incorporate an element of human error. By hard coding a set of potential points of interest there is the possibility of missing a point in time when it would be beneficial to apply a corrective action. Due to the limited testing that was able to be performed it is unknown to what extent this would hinder the search, or whether it was overwhelmed by other elements.

Note that these last two issues are specific to the UPMurphi model and should not discourage the use of a planning approach. However, all these elements combine to demonstrate that a 'full' description of Unit Commitment as a planning problem is not yet feasible. Looking at the reasons for this reveal that it is the numeric sub-problem, attempting to solve Economic Dispatch, which is at the heart of the problem. The complex cost models, convoluted action structure they require and the extra decision points these introduce appear to be the biggest barrier to a successful implementation.

Motivated by this, the following chapter proposes a domain specific Automated Planning system for tackling Unit Commitment. The key feature is separating the two sub-problems, removing the Economic Dispatch dispatch sub-problem from the planning search, instead receiving guidance on optimality from an external solver.

Chapter 6

A Separated Planning Model

6.1 Motivation

The use of planning for Unit Commitment has been predicated on both the temporal structure of the problem and the temporal reasoning abilities of planners, as demonstrated in the literature (see Chapter 4.2). Whilst the temporal part of the solution is the key part, it is intrinsically linked with the numeric aspect used to confirm the cost of a proposed scheduling of units.

Mathematical programming has an indisputable advantage with regards to solving a linear optimisation problem. Whereas the temporal aspects of the problem can be thought of as fitting more naturally into a planning formulation, the numerics fall naturally into a mathematical optimisation framework.

As expected from the review of planning literature in Chapter 4.2 and evidenced in Chapter 5.3 by the planner stalling with even simple problem variants, numerical optimisation is not a strength of planning. Fortunately there is a clear separation in Unit Commitment between the temporal and numeric sub-problems. Supposing an effective separation could be implemented this approach could enable planning to tackle the temporal sub problem and receive guidance on optimisation from an external solver.

Studying the domain from the previous chapter, reveals a clear separation between those actions representing the temporal aspects of the problem, and those representing the numeric aspects. `switch_on` and `switch_off` schedule the units

and `envelope` ensures the actions are run over the same periods as the Timed Initial Fluents representing demand. These three actions therefore represent the temporal aspects of the problem. `ramp_up`, `ramp_down` and `generate` all track the units' output, system supply and attempt to track the cost.

The `ramp_up` and `ramp_down` actions were the real difficulty in scheduling, as they could be applied at almost anytime. Creating a heuristic which meant the planner didn't schedule ramping as late as possible proved problematic, forcing fastest ramping units to be used. To remedy this the `min_timer` function was used in the UPMurphi implementation, but this still had problems.

Similarly the `generate` action was a work-around rather than a natural expression of the problem. In order to track the no-load cost `ramp_up` and `ramp_down` actions were supplemented with a `generate` action to ensure the cost is tracked. Whilst this could be modelled it was inefficient and is likely to have slowed the planner's reasoning down by increasing the number of applicable actions at all points in time.

Furthermore, these three actions all had variable durations, and had to be scheduled to be ran at the exact same times as the `switch_on` and `switch_off` actions. This required more complex reasoning, for simultaneous actions and variable durations. Whilst planning mechanism have been developed to handle these constructs, they are complex and increase solution time. It would therefore appear to be beneficial to remove these three actions from the planning model i.e. remove the numerical optimisation side of the problem.

One key motivation for attempting to model Unit Commitment as a planning problem was the reduced number of actions compared to the number of variables in a MIP formulation. The scale of this reduction and impact it had on search in the previous model was dubious due to the planner performing Economic Dispatch through the `ramp_up`, `ramp_down` and `generate` actions. Removing these actions removes the near infinite number of branching possibilities from reasoning about frequency and duration of those actions. Intuitively this may reduce the reasoning required within the planner.

Clearly these actions cannot be removed unless replaced by another mechanism.

Considering how a basic MIP Branch and Bound solver tackles the problem highlights a parallel to a planning approach which could be exploited. A Branch and Bound solution process typically will solve multiple relaxations of the problem, where a subset of the binary on off variables are fixed, becoming parameters not decision variables, and others are relaxed to become real on [0,1] not integer in {0,1}. Assuming a MILP, gives a relaxation as an LP, which can be solved quickly. The result of this LP informs the MILP search which nodes in the search are more promising than others.

In a typical planning methodology implementing a forward state space search, a state is reached and a heuristic tackles a relaxed variant of the problem. The solution to the relaxation informs the planner which nodes in the search are more promising to expand next. This is a direct parallel of the Branch and Bound process. The sub-problem being handled by the actions `ramp_up`, `ramp_down`, and `generate`, which include the ramping constraints, the outputs and supply are the same that are being solved by the LP in the Branch and Bound process.

It can be seen that in the MIP, these constraints and variables are used to assess the cost and guide the search. This is the same task as a heuristic in planning models. Supposing a dedicated heuristic can be developed to provide accurate guidance on the cost impact of the temporal actions and integrated within the existing temporal reasoning of a planner, an efficient planning methodology which does not suffer from the same problems seen in the model described in Chapter 5 could be developed.

As a result of the poor initial tests running the full Unit Commitment model on UPMurphi, and the above reasoning, the decision was made to pursue a separated model where the Economic Dispatch sub-problem is removed from the model. The planner must now communicate with external solvers to perform dispatch, check state validity, and update a metric to ensure the planner is guided towards optimality. This external solver must be developed and integrated with the planner before performance can be analysed and improved.

Finally, there has been a wealth of work on modelling Unit Commitment as a mathematical programming problem and it would be wise to use some of that work if possible. The second model presented below takes advantage of this vast body of

work in the literature by using existing relaxations.

Below, how this separated model changes from the model described in Chapter 5, is highlighted. Initial thoughts on what is required from the heuristic function, how one would integrate this within a planning system and immediate barriers to implementation are also discussed.

6.2 The Model

In this separated model with Economic Dispatch removed the predicates are reduced to:

```
(on ?u - unit)    (off ?u - unit)
(canSwitchOn ?u - unit)    (canSwitchOff ?u - unit)
(complete)    (demandServiced)    (precondition-to-start)
```

As before predicates `on` and `off` represent the generating unit's online status and are switched with the actions `switch_on` and `switch_off`, shown in Figure 6.1.

Clearly before a unit can come online through `switch_on` it must be offline. The second precondition `canSwitchOn` is a flag which is negated when a unit comes on and only becomes true again with the effect `(at end (canSwitchOn ?u))` of the `switch_off` action. Similarly `switch_off` has a precondition `canSwitchOff` only made true by the effect `(at end (canSwitchOff ?u))` of `switch_on`. In this way `switch_off` must follow the end of `switch_on`. By fixing durations of `switch_on` and `switch_off` to the minimum online and offline durations respectively each unit is forced to be online / offline for at least its respective minimum online / offline period. This is illustrated in Figure 6.2.

The benefit of this action choice is that the constraints are enforced but the actual online and offline times of a unit are not fixed. There are no upper bounds on the time a unit can be left online or offline for as the actions don't negate the precondition they switch. The actions themselves however do have a fixed duration, not expressed as an inequality, making for simpler more efficient reasoning by the planner.

```
(:durative-action switch_on
 :parameters (?u - unit)
 :duration (= ?duration (minimumOnTime ?u))
 :condition ( and
         (at start (off ?u))
         (at start (canSwitchOn ?u)))
 :effect ( and
         (at start (on ?u))
         (at start (not (off ?u)))
         (at start (not (canSwitchOn ?u)))
         (at end (canSwitchOff ?u))
         (at start (increase maxSupply (generationMax ?u)))
         (at start (increase minSupply (generationMin ?u)))))


(:durative-action switch_off
 :parameters (?u - unit)
 :duration (= ?duration (totalOffTime ?u))
 :condition ( and
         (at start (on ?u))
         (at start (canSwitchOff ?u)))
 :effect ( and
         (at start (not (on ?u)))
         (at start (off ?u))
         (at start (not (canSwitchOff ?u)))
         (at end (canSwitchOn ?u))
          ;;
         (at start (decrease maxSupply (generationMax ?u)))
         (at start (decrease minSupply (generationMin ?u)))))
```

Figure 6.1: The two actions responsible for switching the online status of a unit and enforcing the temporal constraints.

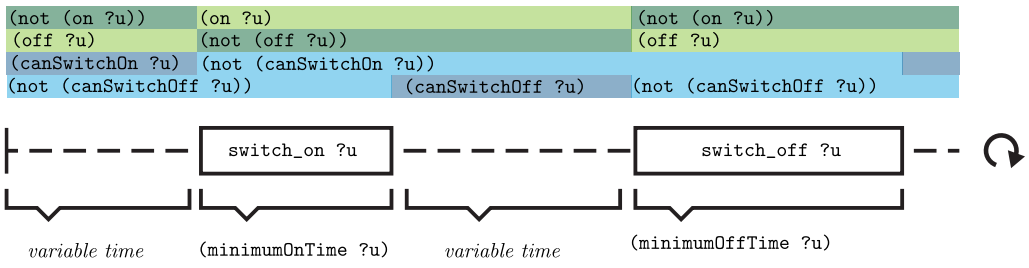


Figure 6.2: The two actions switch_on and switch_off can be scheduled to give any length online and offline periods despite having fixed duration. The temporal constraints are enforced using the predicates canSwitchOn and canSwitchOff

Unlike the original model, supply is handled by the external solver so the exact outputs of the generating units are not considered by the planner. In order to make sure almost all states generated are able to be correctly scheduled by the external solver, the maximum possible output of all generating units is required to be greater than the current demand.[1]

This constraint is enforced by the final temporal action `envelope`, Figure 6.3. `envelope` has a fixed duration equal to the length of the planning horizon and requires the fluent `maxSupply`, updated through `switch_on` and `switch_off`, to be greater than the fluent `demand`, updated through a series of Timed Initial Fluents set in the problem file.

The action contains three more predicates which combine to ensure this action is run for the first x minutes of the planning horizon, assumed to be the same period as the Timed Initial Fluents updating `demand`. The goal requires the two predicates `demandServiced` and `complete` to be true. `demandServiced` is initially false and only set to be true by `envelope`, forcing the planner to run this action. `complete` is initially false and set true using a Timed Initial Literal at the end of the planning horizon. The condition `(at end (complete))` of `envelope` ensures this action is not run too early. In the problem file `(precondition-to-start)` and `at 0.005` not (precondition-to-start)))— coupled with no action setting `precondition-to-start`

[1]This assumes no reserve requirements. Reserve requirements are not being considered in this model and should be built in once a successful solver has been developed.

```
(:durative-action envelope
 :parameters ()
 :duration (= ?duration 1440)
 :condition (and
         (over all (> maxSupply demand))
         (at end (complete))
         (at start (preconditionToStart)))
 :effect (and
         (at end (demandServiced))))
```

Figure 6.3: Action to enforce the system balancing constraint. Predicates are used to ensure this action is scheduled at the same time as Timed Initial Fluents updating the demand.

```
(:action dummy_cost
 :parameters ()
 :precondition ()
 :effect (and (increase totalCost 1)))
```

Figure 6.4: Action to ensure the totalCost fluent, to be updated internally after external Economic Dispatch is performed, is not turned constant in preprocessing

to true means the planner must schedule the action envelope in [0, 0.005). As this action must run and has a duration the entire length of the planning horizon, the constraint `(over all (> maxSupply demand))` is always enforced.

The final action in this model is a dummy action, Figure 6.4. The preprocessing step in POPF, intended for use to solve this problem, removes from consideration any actions that may never be fired and any fluents / predicates which can only remain constant. This dummy action increases the `totalCost` fluent by 1. As the problem file specifies `totalCost` as the metric to minimise this action will never be called by the planner but prevents the preprocessor from setting the variable as static so it can be updated internally once the external dispatch solver completes.

This is a very simple model and removes the need for the planner to consider the

Economic Dispatch optimisation problem which stalled the search in the previous model. The external solver will provide guidance on optimality and the hope is that delegating optimisation to a mathematical programming solver will be the key to unlocking optimality and performance from an AI Planning approach to Unit Commitment.

6.3 The POPF Algorithm

A more formal explanation of how a planning problem is expressed and solved in POPF is presented here. POPF contains some complex concepts having evolved through a family of planners of increasing complexity. The evolution can be thought of as a STRIPS planner, FF [122], extend by METRIC-FF [176], CRIKEY 3 [134] and POPF [177]. The following descriptions are based on those from the original publications relating to each planner.[2]

6.3.1 STRIPS Planning Problems

A basic planning problems is a STRIPS problem[3]. This contains a set of boolean propositions $p \in \mathcal{P}$ and actions which change these propositions $a \in \mathcal{A}$. Define the set off all possible configurations of boolean propositions as $\mathcal{S}$, and as defined in [176, §2] a world state is a set of propositions which are assumed true i.e.

$$s \in \mathcal{S}, \qquad s \subseteq \mathcal{P} \tag{6.3.1}$$

All propositions in the subset $\mathcal{P} \backslash s$ are assumed false. An action is a triple of subsets of $\mathcal{P}$,

$$a := (\rho(a), \mathcal{E}_+(a), \mathcal{E}_-(a)), \qquad \rho(a), \mathcal{E}_+(a), \mathcal{E}_-(a) \subseteq \mathcal{P} \tag{6.3.2}$$

where $\rho(a)$ are those propositions which must be true for the action to be applied, $\mathcal{E}_+(a)$ is the set of propositions which are true after the action has been applied, and $\mathcal{E}_-(a)$ is the set of propositions which become false after the action has been

[2]Other planners and features are omitted for simplicity, to focus only on those features necessary for modelling Unit Commitment

[3]The STRIPS language is named after the STanford Research Institute Problem Solver

applied. $\mathcal{E}_+(a)$ and $\mathcal{E}_-(a)$ are known as the **Add** and **Delete** effects of action a, and $\rho(a)$ as the **Precondition** set.

The effect of an action a applied to a world state s can therefore be defined as:

$$result(s,a) = \begin{cases} s \cup \mathcal{E}_+(a) \backslash \mathcal{E}_-(a) & \rho(a) \subseteq s \\ \text{undefined} & \text{otherwise} \end{cases} \tag{6.3.3}$$

The effect of a sequence of actions $\langle a_1, \ldots, a_m \rangle$ can be defined recursively as:

$$result(s, \langle a_1, \ldots, a_m \rangle) = \begin{cases} result\Big(s \cup \mathcal{E}_+(a_m) \backslash \mathcal{E}_-(a_m), \langle a_1, \ldots, a_{m-1} \rangle\Big) & \rho(a_m) \subseteq s \\ \text{undefined} & \text{otherwise} \end{cases} \tag{6.3.4}$$

The effect of the empty sequence is given by $result(s, \langle\rangle) = s$.

A STRIPS **Task** is given by a 4-tuple ($\mathcal{P}$, $\mathcal{A}$, $\mathcal{I}$, $\mathcal{G}$), where $\mathcal{I} \subseteq \mathcal{P}$ is the initial state and $\mathcal{G} \subseteq \mathcal{P}$ is the goal state, both should be non-empty. The solution to the task is the sequence of actions

$$\pi := \langle a_1, \ldots, a_m \rangle, \qquad \text{such that} \qquad \mathcal{G} \subseteq result(\mathcal{I}, \pi) \tag{6.3.5}$$

If no such sequence of actions exists the problem is unsolvable.

Let each world state $s \in \mathcal{S}$ be represented by a node on a graph. An edge connecting s and s' exists if and only if there exists an $a \in \mathcal{A}$ such that $\rho(a) \subseteq (s)$ and $s' = s \cup \mathcal{E}_+(a) \backslash \mathcal{E}_-(a)$. A search graph can be easily defined, and one can implement basic searches on this graph.

The POPF family uses an **Enforced Hill Climbing** algorithm. Here the graph is not constructed a priori, instead the algorithm iterates through each applicable action in the current state in a breadth first search manner until a state with a lower heuristic value than the current state is found. The action achieving this state is appended to the plan i.e. the graph is expanded along that edge. The algorithm repeats until the goal is reached. If no such action is found the algorithm either fails or a backtracking routine is implemented. In POPF, if Enforced Hill Climbing fails either a Best First or A* search is used, both of these require more memory and time but are more likely to find a solution.

The basic heuristic of the POPF family is based on a forming Relaxed Planning Graph which can be solved quickly. As an example, consider the following definition [176, §2.1]:

Definition 6.3.1. Assume a STRIPS Task ($\mathcal{P}$,$\mathcal{A}$,$\mathcal{I}$,$\mathcal{G}$). Define the relaxation of an action a as

$$a+ := (\rho(a), \mathcal{E}_+(a), \emptyset) \tag{6.3.6}$$

The relaxation of ($\mathcal{P}$, $\mathcal{A}$, $\mathcal{I}$, $\mathcal{G}$) is then ($\mathcal{P}$, $\mathcal{A}^+$, $\mathcal{I}$, $\mathcal{G}$) where $\mathcal{A}^+ := \{\ a^+\ :\ a \in \mathcal{A}\ \}$. A sequence $\langle a_1, \ldots, a_m \rangle$ is a relaxed plan for ($\mathcal{P}$, $\mathcal{A}$, $\mathcal{I}$, $\mathcal{G}$) if $\langle a_1^+, \ldots, a_m^+ \rangle$ is a plan for ($\mathcal{P}$, $\mathcal{A}^+$, $\mathcal{I}$, $\mathcal{G}$)

From a state s, all states in the **Relaxed Planning Graph** (RPG) $\mathcal{R}(s)$ (the graph of all states reachable by all relaxed actions) will form a non-decreasing subset of $\mathcal{P}$, as no proposition will ever be falsified. Thus the distance to the goal can be estimated as the number of actions in the relaxed plan.

6.3.2 Crikey and Crikey 3

CRIKEY is a temporal planner that can reason about time and duration but not about numeric properties [131]. Crikey is particularly strong in that it reasons correctly about required concurrency. CRIKEY also supports actions with variable durations.

There are two broad situations in which required concurrency can arise: interactions between activities in the domain or a deadline which forces actions to be compressed. Concurrency from compressing actions can be found be creating a non-temporal plan and then scheduling the actions afterwards but as detailed in [191, §2], in many cases this will be an inefficient way to generate the plan. Required concurrency from interactions between activities in the domain means that the planning cannot be separated from the scheduling and a new representation of durative actions should be built.

CRIKEY 3 builds on the CRIKEY framework adding support for numeric quantities and a more efficient model for temporal constraints. The updated state representation and state succession are presented below.

In the initial planning model for Unit Commitment there is both required concurrency (the `generate` and ramping actions concurrently with the `switch_on` action) and variable durations (amount of time to ramp for), which complicates reasoning. A second model does not contain these features so an explanation of how they are handled is omitted.

Temporal Representation

In CRIKEY and CRIKEY 3 a **Simple Durative Action** da is defined as a tuple

$$da := (\underbrace{C_{\vdash}, C_{\leftrightarrow}, C_{\dashv}}_{\substack{\text{start, overall and} \\ \text{end conditions}}} , \underbrace{\mathcal{E}_{+,\vdash}, \mathcal{E}_{+,\dashv}}_{\substack{\text{start and end} \\ \text{add effects}}} , \underbrace{\mathcal{E}_{-,\vdash}, \mathcal{E}_{-,\dashv}}_{\substack{\text{start and end} \\ \text{delete effects}}} , \underbrace{\Delta}_{\text{duration}}) \tag{6.3.7}$$

which form a set of all durative actions

$$da \in \mathcal{A}_{\Delta} \subseteq \mathcal{A} \tag{6.3.8}$$

The notion of durative actions requires an introduction of a time scale. During search this is represented as a series of temporal relationships. Actions are fixed to a definite timeline once planning is complete. The following definitions formalise these ideas.

Definition 6.3.2. A durative action is split into two **Snap Actions**

$$a_{\vdash} = (C_{\vdash}, \mathcal{E}_{+,\vdash}, \mathcal{E}_{-,\vdash}) \tag{6.3.9}$$

$$a_{\dashv} = (C_{\dashv}, \mathcal{E}_{+,\dashv}, \mathcal{E}_{-,\dashv}) \tag{6.3.10}$$

Definition 6.3.3. A **Temporal Constraint** is a constraint of the form

$$x_a - y_b \ \{\leq, <, >, \geq\} \ d \tag{6.3.11}$$

where x_a, y_b are start or end timestamps of snap actions a, b and d is a cap on their separation from durative action conditions. Temporal ranges and equalities can be captured through conjunctions of temporal constraints. We denote a preceding (succeeding) b with $a \prec (\succ)\ b$ and call this a **Partial Ordering**.

Definition 6.3.4. A collection of temporal constraints is known as a **Simple Temporal Network** (STN). An STN is said to be **consistent** if there exists a consistent temporal embedding (see below).

Definition 6.3.5. A **Temporal Embedding** is a map $f : \mathcal{A} \to \mathbb{R}$ where the section of $\mathbb{R}$ mapped to represents the times at which the actions will occur. A **Consistent Temporal Embedding** for the instant actions π in an STN S is a map such that

$$a \prec b \implies f(a) < f(b) \qquad \forall\ a, b \in \pi \tag{6.3.12}$$

and for all durative actions, $da \in \pi$, with snap actions, $a_\vdash, a_\dashv$,

$$f(a_\dashv) - f(a_\vdash) = \Delta \tag{6.3.13}$$

A plan, π, is now an ordered sequences of instantaneous and start / end snap actions (which can be resolved to durative actions) for which the temporal constraints form a consistent simple temporal network.

World State Representation

A world state is now required to encode the information about temporal constraints, and also any `over_all` conditions of the durative actions, $C_\leftrightarrow$. Define a durative triple

$$e(da) := \langle a_\vdash, i, \Delta \rangle \qquad \text{where} \qquad \begin{cases} a_\vdash \text{ is a start snap action of } da \\ i \in \{1 \dots |\pi|\} \\ \Delta \in \mathbb{R} \end{cases} \tag{6.3.14}$$

to be a triple of a start snap action, the step in the plan at which this snap action was added, and the duration of the action.

A state in Crikey 3 is defined to be a triple of all currently true propositions, a set of durative triples of the open durative actions whose start snap actions but not end snap actions have been added to the plan, $\mathcal{A}^o_\Delta$, and a set of temporal constraints:

$$s = \langle F, E, T \rangle, \qquad \text{where} \qquad \begin{cases} F \subseteq \mathcal{P} \\ E = \{e(da)\}_{da \in \mathcal{A}^o_\Delta} \\ T = \{x_a - y_b\ \{\leq, <, >, \geq\}\ d\} \end{cases} \tag{6.3.15}$$

State Progression

State progression in this temporal setting is to apply either an instantaneous action, or a start / end snap action. An instantaneous action a is applicable if its preconditions $\rho(a) \subseteq F$ and its effects do not invalidate any of the `over_all` conditions of any open durative actions. Supposing the action is applicable the result is

$$s' = result(a, s) = (F', E, T) \qquad \text{where} \qquad F' = (F \cup \mathcal{E}_+) \backslash \mathcal{E}_-$$

Appending a snap action to a plan has the following effects

$$s' = result(a_\vdash, s) = (F', E', T) \qquad \text{where} \qquad \begin{cases} F' = (F \cup \mathcal{E}_{+,\vdash}) \backslash \mathcal{E}_{-,\vdash} \\ E' = E \cup \langle a_\vdash, i, \Delta \rangle \end{cases}$$

$$s' = result(a_\dashv, s) = (F', E', T') \qquad \text{where} \qquad \begin{cases} F' = (F \cup \mathcal{E}_{+,\dashv}) \backslash \mathcal{E}_{-,\dashv} \\ E' = E \backslash \{e \in E | a_\dashv \text{ corresponds to } e.a_\vdash\} \\ T' = T \cup \{t(i) - t(e.i) = e.\Delta \\ \qquad\qquad\qquad | a_\dashv \text{ corresponds to } e.a_\vdash\} \end{cases}$$

The tests for applicability and state representations now require more memory.

Temporal Search

Search in Crikey is as before, where the set of all applicable actions now includes the set of all applicable start and end snap actions as detailed above. The heuristic in Crikey 3 and Popf is complex and is replaced in the proposed procedure for Unit Commitment, so is not detailed here. Relevant information can be found in [177, §5]. The STN is resolved by enforcing that each step in the plan is separated by an arbitrary minimum amount, $\epsilon > 0$, i.e.

$$T_f = T \bigcup_{i>1} \{t(i) - t(i-1) \geq \epsilon\} \cup \{t(1) \geq 0\}$$

and solved using standard techniques.

In Popf the search is still forward chaining, but incorporates ideas from partial order planning. This has the benefit of better reasoning with deadlines, or Timed Effects (see below) [177]. The concept of partial ordered planning is that actions

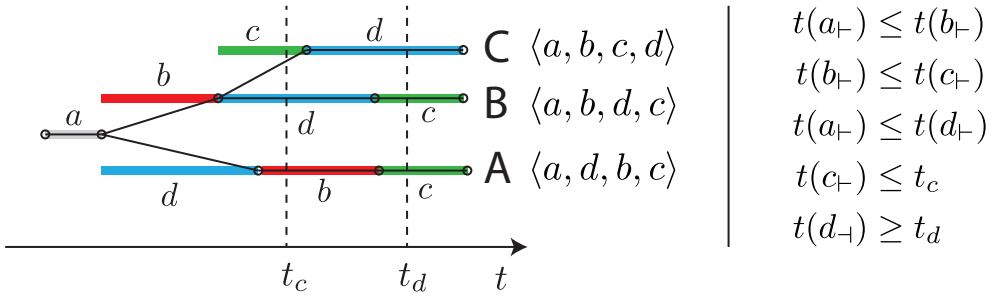


Figure 6.5: Schematic illustrating the benefit of partial order planning. Suppose a, b, c and d are all durative actions with the temporal constraints as in the figure. In a total order planning algorithm the planner may arrive at point A before considering the Timed Initial Literals enforcing the final two constraints. It would then have to back track, possibly attempting point B, before arriving at the only valid solution, C. In a partial order planning scheme, the temporal order of d in relation to b and c is not set until the end of search when all constraints will have been added to the STN, thereby avoiding backtracking and unnecessarily extending search.

are not committed to the order they are initially added in the plan. In all search methods mentioned above, if an action is added to the search at the third step, no new action can change its timestamped order, it will always be executed third. In partial order planning this is not the case.

Specifically in POPF, the planner is allowed to demote and promote actions already added to the plan to a lower or higher timestamped order. Consider a plan consisting of 4 actions a, b, c, d, such that a must precede b, which must precede c, and d must occur after a but does not interact with either b or c. The orders $\langle a, d, b, c\rangle$, $\langle a, b, d, c\rangle$ and $\langle a, b, c, d\rangle$ are all valid. A partial ordering scheme like that used in POPF will not commit the order of these actions until the end of the search.

This is beneficial as it may be that later actions require a late implementation of d and thus a total order planner would have to backtrack past either of the first two orders in order to find the correct later scheduling, as illustrated in Figure 6.5. POPF would simply add the later constraint to the STN when the requirement became apparent, preventing unnecessary backtracking.

6.3.3 Supporting Numerics

The POPF family incorporated numeric support from CRIKEY 3 onwards, however much of the support is very technical and supports complex language constructs stripped out of the Unit Commitment problem in the second formulation. Avoiding this costly reasoning allows the planner to focus on the temporal structure of the problem. Presented here is an introduction to how the numerics are handled in the POPF family, omitting technical constructs not needed for handling the second Unit Commitment variant.

Analogous to the set of propositions $p \in \mathcal{P}$, define a set of continuous real valued variables $v_i, \{v_i\} := \mathcal{V} \in \mathbb{R}^n$, where n is the number of fluents. The world state is now appended, becoming a 4-tuple

$$s = \langle F, E, T, V \rangle$$

Numeric effects supported by POPF are discrete pre-determined effects, situation dependent discrete effects, and, as of COLIN, non-discrete continuous linear effects. Define a **Numeric Effect** as

$$e(v_i) := v_i \quad \{=, = v_i +, = v_i - \} \quad x \quad \{+, -, \times, \div\} \quad y \tag{6.3.16}$$

where x, y can be other variables or constants, thus assigning ('='), increasing ('$= v_i+$') or decreasing ('$= v_i-$') the fluent variable v_i by the value of the right expression. An action, durative or non-durative, is appended with numeric effects for some subset of fluents, $\tilde{\mathcal{V}} \subseteq \mathcal{V}$

$$a := (\rho(a), \mathcal{E}_+(a), \mathcal{E}_-(a), \mathcal{E}_v(a)) \qquad \text{where} \qquad \mathcal{E}_v(a) = \{e(v_i)\}_{v_i \in \tilde{\mathcal{V}}} \tag{6.3.17}$$

CRIKEY 3 was the first planner to support reasoning about numeric effects with required concurrency [191]. Resources which are only available during a durative action's execution are not lost in the snap action and durative triple formulation, but would be in the compressed durative action formulation of other earlier planners.

Discrete changes update the world state as propositional changes do, and can also be placed in the start and end effects of durative actions. Continuous changes are those which are specified using the `#t` construct. COLIN, which followed CRIKEY

3, supported continuous changes and encodes them into an LP which is solved in conjunction with the STN to ensure the temporal numeric interactions are valid. This process is detailed in [133, §8].

Timed Effects

Two important constructs in the Unit Commitment problem which remain in the second model are Timed Initial Literals, to model initial online / offline periods, and Timed Initial Fluents, to model the demand profile. These are handled using dummy actions. These dummy actions are added to the plan at a step i by amending the facts F or fluents V and appending the temporal constraints

$$T' = T \cup \{t(i) - t(a_0) = \tau\}$$

where τ is the time of the timed effect, a_0 is a dummy action representing the start of the planning period ($t(a_0) = 0$) and the dummy action is applicable only once earlier TILs have been appended to the plan.

Significant modifications to the Relaxed Planning Graph formulation are necessary to reason about the temporal, numeric and timed effects features. In the second planning model presented here the heuristic is domain specific and not based on an RPG so their discussion is omitted, but can be found in [133, §9].

6.4 Extending the POPF Algorithm for Unit Commitment

Based on the above discussion the search process can be described as follows.

1. Construct the initial world state to encode just the facts that are initially true and the fluents assigned default values as given in the problem. There are no initially open durative actions and no temporal constraints giving an initial state as

   $$s_0 := \langle F_0, \emptyset, \emptyset, V_0 \rangle$$

   The initial plan is also empty

   $$\pi_0 = \langle\ \rangle$$

2. Define $i = 0$. $s_i \leftarrow s_0, \pi_i \leftarrow \pi_0$.

3. Calculate the heuristic of the current state, $h_i \leftarrow h(s_i)$. In POPF, the heuristic function is an estimated distance to the goal state as found from the Temporal Relaxed Planning Graph.

4. From the Temporal Relaxed Planning Graph, construct the set of useful actions as those in the relaxed plan from the current state whose preconditions are satisfied in the current state, i.e.

$$\mathcal{A}_h \leftarrow \{\ a \in \pi_{TRPG}\ :\ \rho(a) \text{ or } C_{\vdash}(a) \text{ or } C_{\dashv}(a) \subseteq s_i.F\ \}$$

5. Iterate through all applicable actions in a breadth first manner as follows. For $a \in \mathcal{A}_h$

   (a) If a is not applicable, advance a. Go to Step 5a.

   (b) Apply the action a i.e $s' \leftarrow result(a, s_i)$

   (c) If the new STN, T', is not consistent, advance a. Go to Step 5a.

   (d) Evaluate the heuristic. $h' \leftarrow h(s')$.

   (e) If $h' < h_i$,

$$s_{i+1} \leftarrow result(a, s_i)$$
$$\pi_{i+1} \leftarrow \langle \pi_i, a \rangle$$
$$i \leftarrow i + 1$$

   Go to Step 3.

   (f) Else advance a. Go to Step 5a.

The separated model discussed in Section 6.2 requires a custom heuristic to check state validity and calculate a heuristic. To achieve this there are two key areas to address, the heuristic and the state validity. These are performed in Steps 3, 5d and 5c.

Supposing the planning problem wholly encodes propositional and numeric validity in its applicability test then Step 5c necessarily removes any invalid states. In

this separated model of Unit Commitment a further test is required. The envelope action only requires that the maximum possible supply to be greater than demand, this does not imply the current online / offline schedule can create a valid dispatch schedule.

If the current schedule cannot be dispatched in such a way that demand is met, most likely due to ramping constraints not encoded in the model, the state should be dismissed as invalid. This is also the place to increase complexity when looking to increase the realism and complexity of this model. Network constraints could be built into the checking procedure increasing the realism of the problems tackled. This checking is done assuming all temporal decisions have been made, so these parts of the problem are simpler than building them into a MIP formulation.

State validity can be efficiently checked as part of the heuristic function, the requirements of which are discussed in detail below.

6.4.1 Heuristics

Running this model on POPF unchanged is not representative as there is no notion of production cost in this model. The first step to implementing this model is to design and construct a domain specific heuristic. This would calculate the exact cost of scheduling the temporal decisions made so far, and the expected cost to go, the sum of which would be used as a heuristic. Use of a complex heuristic without implicit increases in solution time is made possible due to the fewer decision points in the search resulting from the removal of ramping and `generate` actions.

The complications include:

1. Grounding the temporal decisions made so far
2. Efficient modelling and solution of the dispatch assuming the grounded temporal decisions
3. Efficient estimation of the cost to go
4. Possible extraction of helpful actions

Performing Dispatch for Cost So Far

Of the above complications, (2) is perhaps the simplest. The time horizons for this will be as far through the planning period as the planner has searched. This will almost always be much less than the full horizon and so solution times should be short. One could simply take the MIP model in Appendix A.1 and have the binary variables fixed according to the actions already scheduled in the plan. This then becomes an LP with the complex constraints (A.1.6) - (A.1.18) removed. This also means that the final stages of the plan will perform the dispatch for the problem, the solution of which can be outputted alongside the plan.

The downside of this is in the sheer number of LPs to be solved. Potential ways to overcome this would be to look back only up to 6 hours. If validity is to be checked at each action point then once a plan spans 18 hours, the validity of the first 12 hours of the plan should be confirmed and it is unlikely any action will break that validity. There is a chance of breaking the validity given that the partial ordering can alter the temporal ordering of the actions but intuitively this seems unlikely.

Another option is to reduce the amount of checking to be done. Simplistically one could use the algorithm for calculating the 'cost to go' part of the heuristic (see below) and only run an LP either at fixed intervals in plan steps or fixed intervals throughout the planning horizon i.e. every 5 action steps or first time the plan exceeds 3, 6, 9, 12 etc hours. Of course the checking schemes should be varied and studied to see what tradeoffs can be made at this point in the algorithm.

It is likely that a more simplistic heuristic would result in poorer guidance, although this would have to be tested to be confirmed. A compromise may be to choose the next candidate greedily, only comparing the 'cost to go' part of the heuristic in Step 5d.

Given the Enforced Hill Climbing algorithm used in the first pass of POPF's search, where the comparisons are between all actions stemming from the same partial plan, this approximation would be equal to the full heuristic. State validity could then be checked at intervals as discussed above.

This 'cost to go' only approximation would probably prove quite poor in an A* search situation, the second pass of POPF's search. It is likely the behaviour would

be very similar to that seen in UPMurphi, where plans earlier in the horizon have a much larger cost to go than those towards the end. It is likely the planner would again be inherently drawn towards choosing those plans with a later span over those with better earlier decisions.

One would infer that for fast searches, where a feasible solution found quickly is desirable, Enforced Hill Climbing combined with the approximation checking validity at intervals would provide a considerable time saving by dramatically reducing the number of LPs to solve. In a more optimisation focussed framework however, A* with the more accurate cost so far provided by an LP would seem to be the better approach. These approaches are illustrated in Figure 6.6. Full testing of both of these situations should be done before any conclusions can be drawn.

Moving from the theory of this part of the algorithm to its implementation in code, there is work to be done to extract the model in the preprocessing stage of the code to reduce communication overhead. Much of the model (the units and their characteristics, the demand profiles etc) is static throughout the plan and should not be transferred between solvers each time if possible.

A close study of the CPLEX or COIN external APIs is required to ensure communication between solvers does not disproportionately lengthen the solution time. One possibility would be not to consider each applicable action singularly, but batch process calculating the heuristic for 5 different actions. This may reduce the effect of any bottlenecks caused by communications but may also lengthen the solution process as an Enforced Hill Climbing algorithm has the benefit of quickly moving up any path which appears better.

The possibility of warm starting the LP based on previous solutions should also be considered and any solution time reduction weighed against the memory overhead required in storing previous solutions. There may well be a great reduction in solution time as most queries will only vary slightly from state to state having only advanced a little further in time.

This is also an easy place to implement parallelism by combining batch processing of heuristic calculations and warm starting the LP solvers. Warm starting from previous solutions would intuitively bring a lot of performance with little effort on

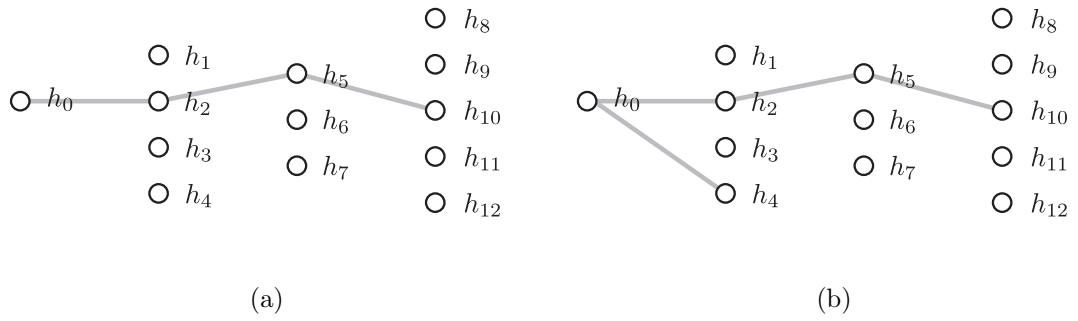


Figure 6.6: Schematic illustrating two differing search frameworks. Suppose the search is at state 0, then if $h_1 < h_0$ search advances to state 1. Otherwise h_2 is calculated and the search continues in this way. There is no cost associated with reaching the initial state so h_1 and h_2 are the costs of reaching their respective states and the expected cost to go from that state. Similarly, suppose we are expanding from state 2, then h_5, h_6 and h_7 are all the cost of reaching state 2, plus their action costs and remaining cost to go. Given that the planning model does not handle costs, and the cost of the switch on action is not just the cost of bringing a unit online, but also the cost of its expected contribution to generation, these specific action costs cannot be singled out. They are however included in the simple cost to go heuristic outlined in the body of the text. This is again the case when expanding state 5, assuming $h_5 < h_2$. $h_8 - h_{12}$ are all the cost of reaching h_5 plus the expected cost to go once the respective actions have been applied. Thus in an Enforced Hill Climbing framework only the expected cost to go is needed as the cost so far will be the same for each state being compared. Contrastingly, in an A* search the cost to go from state 10 will be much less than from state 4, but the cost so far will be higher, as it is further along the timeline. As these will be compared in an A* search, using only the cost to go will be misleading, as it was in UPMurphi. Instead both the cost of reaching that state and the expected cost to go should be included in the heuristic to give a fairer comparison.

the programmer's behalf as commercial parallel solvers are readily available and little planning code would be altered.

These implementation issues are all avenues to consider once complication (1) has been successfully addressed.

Estimating Cost To Go

Complication (3) is striking a balance. In essence the cost to go is the cost of scheduling for the remainder of the planning horizon, which is simply Unit Commitment where the decisions made so far are committed and the planning horizon is shorter. The balance is in choosing an algorithm which is very fast but accurate enough to be representative.

It is not quite a case of choosing from the existing portfolio of Unit Commitment algorithms. Many important decisions in the Unit Commitment planning horizon are to be made around the time of the morning ramp. The Cost to Go at this point is still over 15-18 hours and so the problem is still reasonably large. Even older priority list methods implemented techniques involving dynamic programming to improve their accuracy [25].

Building solvers for these techniques into the heuristic would be both complex and inevitably result in suboptimal algorithms and implementations, and possibly programmer error. Using another external solver would introduce more communication overheads and reduce readability and maintainability of the code. For practicality reasons it is therefore better to use a very simplistic algorithm for the heuristic, which will inherently be quicker to solve and easier for the researcher to implement, debug and maintain.

A priori it is also unclear what the effect of various heuristics would be as problems such as this have received less attention in the literature than planning problems using more conventional Relaxed Planning Graph approaches. Thus, it is best to test a wide portfolio of heuristics to study what advantages and disadvantages each has. Such testing facilitates informed decisions on which routes to pursue further for tackling larger, more realistic problem instances.

For Unit Commitment as a heuristic we have the freedom to do things such as

- estimate for a shorter period
- decrease the granularity of the demand profile either statically (only use 90 minute discretisation) or dynamically (for the next 3 periods use 30 minute granularity, for the following 3 use 60 minute etc.)
- remove constraints for simplicity

It should be clear that the impact of each of these relaxations on different heuristics is unknown. The criteria for the algorithm as a heuristic is to provide good guidance. In essence this is just clear differentiation between which solutions are better, or more promising, than others. The solution is not required to be a truly realistic and accurate assessment of the cost to go, just indicate situations which are more promising than others.

When discussing the Temporal Relaxed Planning Graph used in Crikey 3, [191], it is emphasised that the new heuristic retains planner completeness, which requires a planner to never incorrectly remove a state believing it to be either a dead end or invalid. It is not immediately clear that the relaxation of the Unit Commitment problem and the proposed heuristics discussed here retain this property. A more in-depth study is required to assert this.

Many approaches will have to be attempted and trialled over a variety of problem instances to determine which heuristic gives the best balance between performance and accuracy.

A first attempt presented here is a very simplistic approach. A 'greedy' deterministic Unit Commitment algorithm can be as in Figure 6.7. In essence this algorithm chooses the next unit to deploy based purely on having the lowest marginal cost. This basic heuristic does not consider the ability to bring a unit offline, minimum online / offline periods or ramping rates but does provided an estimate for the cost to go and is easy to encode efficiently. It is not too complex to extend the algorithm in Figure 6.7 to include the above oversights, see Appendix B.4.

1. Split the set of generating units into two sets, $\mathcal{G}, \mathcal{C}$, for those currently online (**G**enervating) and those currently offline (**C**ool). Order these sets by marginal cost such that $\mathcal{G}_0$ is the lowest marginal cost of the units in $\mathcal{G}$ and $\mathcal{C}_0$ is the lowest marginal cost in $\mathcal{C}$.

2. Let

   - $D(i)$ be the demand for period i
   - $g(u, i)$ be the output from each generating unit
   - $S(i) = \sum_{u \in \mathcal{G}} g(u, i)$ be the total output
   - $\delta_i = S(i) - D(i)$ be the power deficit

3. Let $i \leftarrow 1$ be the first period.

4. Set $g(u, i) \leftarrow G_{\min}(u)$ for all $u \in \mathcal{G}$. Update $S(i)$ and δ_i. Set $u \leftarrow 0$.

5. *Assign Outputs:*

   $$g(u, i) \leftarrow \max\Big[\min(G_{\max}(u), \delta_i), G_{\min}(u)\Big]$$

   Update $S(i)$ and δ_i.

6. If $\delta_i \geq 0$ { if $i = N$ { stop } else { $i \leftarrow i + 1$, Go to Step 4 } }.

7. If $u < |\mathcal{G}| - 1$ { $u \leftarrow u + 1$, go to Step 5 }

8. *Switch On Additional Units:*

   (a) Remove $\mathcal{C}_0$ from $\mathcal{C}$ and insert into $\mathcal{G}$. Let v' be its sorted index in $\mathcal{G}$.

   (b) Set $g(v', i) \leftarrow \min(G_{\max}(v'), \delta_i)$. Update $S(i)$ and δ_i.

   (c) If $\delta_i \geq 0$ { if $i = N$ { stop } else { $i \leftarrow i + 1$, Go to Step 4 } } else { Go to step 8 }

Figure 6.7: A first attempt at a heuristic which determines the units to deploy next in order of marginal cost.

Helpful Actions

Implemented in the POPF family since FF, the reduction of the set of actions over which to test for successors from all of those that are deemed applicable to a 'helpful' subset has been demonstrated to improve performance [122]. It therefore seems appropriate to consider how helpful actions could be extracted from a relaxed Unit Commitment solution, given that the Relaxed Planing Graph framework has been removed.

Initial thoughts would be to restrict attention to those units which are used in the relaxed solution, exactly as POPF determines helpful actions in a temporal setting. The downside of this is that the proposed Unit Commitment heuristic does not have the sophistication of the Temporal Relaxed Planning Graph in POPF and intuitively would present a less considered, more restrictive subset of actions than should be used at this early stage in development.

As helpful actions are primarily to speed up reasoning and do not alter plan accuracy, their consideration should be brought in only once a heuristic providing accurate guidance has been developed.

Grounding the Simple Temporal Network

The final difficulty mentioned, (1), is translating the actions in the plan thus far to concrete binary variables.

Currently the time reached in the planning horizon is determined by the next upcoming Timed Initial Fluent. There has not been any work in this project on translating the actions added to the partial plan so far into fixed time periods for use in calculating the cost so far.

One option could be to resolve the STN as is done at the end of planning, however this will likely be very time consuming. A possibility of warm starting the STN solution may be available but has not been investigated. For initial testing this may not be a problem if the Enforced Hill Climbing search is used in conjunction with the simple heuristic discussed above. This allows for delaying dispatch until the STN is resolved once planning is complete. This has not been tested and will likely contain many issues in itself but is a starting point from which the model can

be developed.

6.5 Next Steps

This chapter proposed a model for solving Unit Commitment using Automated Planning. It removes many of the causes of the difficulties seen in the first model presented in Chapter 5.4. The optimisation which stalled the search in the previous model is removed from the planner's consideration, leaving a temporal problem constrained by numerics.

It is hoped this model coupled with the discussions on how and why it has been constructed in this way will form a good starting point for other researchers looking to model Unit Commitment with Automated Planning. The discussions on possible theoretical areas which could lead to poor performance, and immediate challenges to implementation should begin to form a road map for how to develop this model further.

To clarify this road map, the first requirement is to ensure the simplistic heuristic in Figure 6.7 has an effect on the plan. Initial tests indicated that the resolution of the STN was not correctly influenced by this heuristic. As illustrated in Figure 6.8 the `switch_on` actions were arbitrarily assigned to either end of the feasible time scale. If the wrong end happened to be assigned the switch would occur at the wrong time. This led to some cases appearing as if the heuristic had no effect whilst the real cause was not the heuristic but the STN solver. Weighting the resolution of the STN to the correct boundaries, thereby consistently ensuring the heuristic is utilised, should be a first step towards improving this model.

Next, improving the cost to go heuristic should be considered. Whilst A^* framework discussed above can provide more accurate guidance, having just the cost to go could still be effective in the Enforced Hill Climbing framework. It is hoped that the extra complexity discussed previously can all be incorporated into the 'greedy' heuristic and give more credibility to the heuristic from a Power Systems Engineering perspective.

An assessment of Enforced Hill Climbing with just Cost to Go against A* with

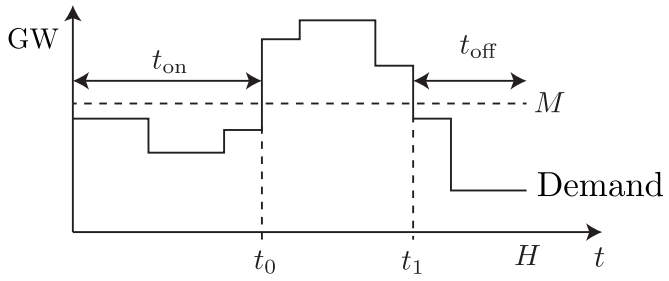


Figure 6.8: Schematic Illustrating the STN resolution problem. Suppose the current maximum supply is M, and a single unit u can be brought online at any time in the planning horizon and will cover the shortfall. The Timed Initial Fluent represented by t_0 will add the constraint $t_{\text{on}} = t(\text{switch\_on}(u)) \leq t_0$ to the STN. Similarly the heuristic would indicate that u can be switched off once the Timed Initial Fluent represented by t_1 has been activated, adding the constraint $t_{\text{off}} = t(\text{switch\_off}(u)) \geq t_1$ to the STN. This gives the following regions for the switch_on and switch_off actions: $t_{\text{on}} \in [0, t_0]$ and $t_{\text{off}} \in [t_1, H]$. The optimal solution is $t_{\text{on}} = t_0$ and $t_{\text{off}} = t_1$, i.e. a switch_on should be resolved to the end of its feasible region, and a switch_off to the start of its interval. The STN solver within POPF should be influenced to ensure this occurs correctly.

Cost so Far + Cost to Go should follow an accurate Cost to Go heuristic. There may be a increase in accuracy using the A* approach but the cost of grounding the STN and solving either LPs or other dispatch algorithms on the solution time should be investigated. Perhaps both avenues could be developed. When rapid re-calculation of committed units is required, in the event of system failures for example, a fast Enforced Hill Climbing framework could be used. In Day-Ahead situations where accuracy is more important and solution times can be longer, and A* framework could be more applicable.

Once these basic principles have been address the model should be pushed for efficiency. Efficient communication. opportunities for efficient solutions through warm starting sub-algorithms, and preprocessing to remove unnecessary overhead at each iteration, should all be investigated to ensure the basic algorithm achieves its potential and is not slowed by implementation details, rather than algorithmic details.

Long term goals should include more realistic Unit Commitment models, such as reserve setting, large uncertainty in net demand from renewables, and controllable loads. It may be that more realistic modelling features can all be built into the state checking process. Supposing the planner can reason about the overall temporal structure efficiently, sub-algorithms and external solvers for the heuristic may be able to perform validity checking under more realistic constraints, but in less complex formulations than when incorporated into a MIP model. This may result in an overall efficiency gain.

Planning under uncertainty has received much attention in the literature. Due to the lack of off-the-shelf planners that could be effectively used to tackle basic Unit Commitment, it is necessary to build a solid foundation for simpler instances. From this foundation techniques on uncertainty can be brought in, however it is beyond the scope of this project.

Chapter 7

Conclusions

7.1 Motivations

Unit Commitment is an integral problem to the power systems industry, crucial for its secure and economic daily operation. For this reason it has received much attention from industry and academia for many years. Major changes are underway in power systems around the world, altering fundamental operational paradigms. Commitments to increasing the amount of renewable energy used for generation necessitates an increase in Variable Generation. As penetrations increase, its contributions must be incorporated into the Unit Commitment problem.

It is widely accepted that a failure to compensate for the variability in this generation will either lead to curtailment of renewable sources, removing the benefit of incorporating it, or more units online for reserves, greatly increasing the cost of generation for that period. Instead the variable generation must be accounted for and thermal generation scaled down accordingly. The stochastic nature of this generation means the contribution to supply from renewable sources is uncertain, especially in the day-ahead time frame. In order for the system to remain secure this uncertainty must be accounted for and any shortfalls from overestimating renewables hedged against.

Quite how this can be accomplished is not known. Current proposals are varied, and further research is a major focus of industry and academic attention in relation to Unit Commitment right now. Despite high levels of attention, industrial scale ap-

plications have not been sufficiently demonstrated and academic approaches remain laboured.

The vast majority of the literature on Unit Commitment uses a MIP formulation or other form of time discretisation. Automated Planning is a relatively new field of Informatics expressing problems in a different way. Whilst there has been less focus on handling complex mathematical optimisation style problems, a category which Unit Commitment falls into, it is very well suited for expressing real-world problems with interesting temporal constraints and complexities, a category which Unit Commitment also falls into.

The object-action formulation of planning problems is well suited to the temporal aspects of the Unit Commitment problem. A fine discretisation of time is required to accurately model the impact of variable generation, necessitating a large number of binary decision variables in a MIP formulation. Contrastingly there are only a few times in the planning horizon when a units online status switches. This contrast in granularities has a large impact on the scale of the MIP problem but does not affect the complexity of the planning domain.

Automated Planning has not received the same level of attention as Mathematical Programming approaches and so does not have the same breadth of expertise demonstrated throughout the literature. Despite this there have been successful implementations of complex real world problems being tackled and implemented in industry implying planning is a viable avenue for research.

Being a relatively new field has the advantage for researches that there are perhaps more avenues to explore than areas which have seen much more sustained attention. Unit Commitment appears naturally suited to a planning approach and the planning community has demonstrated an interest in tackling other problems in Power Systems Engineering. Recent attention in the community has centred on complex temporal problems and handling uncertainties.

Following research such as that presented in Chapter 6.4 this problem could be tackled by Automated Planning and a successful implementation would provide renewed evidence for the promise of numerical reasoning within a complex temporal domain through planning. Academic conferences and events within the planning

community have stressed the desire to tackle real world problems. Unit Commitment is a highly important real world problem that could return attention in the community to real world problems with a focus on numerical optimisation.

7.2 Contributions

This thesis has endeavoured to present Unit Commitment as an integral problem to the power system industry, for which unknowns and open questions remain. There is much interest in both academia and industry, however there is not always a correlation between the latest advances in academia and uptake in industry. With this in mind initial work on creating an Automated Planning model for Unit Commitment has been presented, outlining where benefits over existing methodology may come from. This sections details the outcomes from these areas of study.

7.2.1 Study of Metaheuristics: Lessons Learnt

As discussed in Chapter 2.3, its was around 1999 when Branch and Bound began to gain widespread popularity for Unit Commitment. Over the following decade most large System Operators made use of the formulation. Throughout the previous decade however Meta-Heuristic methods for Unit Commitment such as Genetic Algorithms [93], Tabu Search [88], and Simulated Annealing [77] were all demonstrating viability if not proven performance (see Chapter 3 for a complete discussion). Despite continued academic interest in the field uptake in industry has not been seen.

As this thesis attempts to provide initial discussions on Automated Planning for Unit Commitment, an approach not seen in the literature thus far, it is important to analyse why the academic literature on Meta-Heuristic approaches has not received industry interest. There has been much success in applying many different Meta-Heuristics to simple instances of the problem. There have however been key shortcomings in developing those systems to meaningful scales, and in presentations of competitiveness.

Tangible demonstrations of practical ability through realistic model formula-

tions and real world test systems were stated as key factors missing from much Meta-Heuristics work. Unit ramp rates are easily defined in a MIP formulation (see Appendix A.1 for example) but many Meta-Heuristics approaches did not consider them. Key reasons for this were the complexity in generating feasible solutions during search under these constraints and increases to the solution time. Modern work in certain sub-fields tackled these issues but often retained the recurring oversight of not comparing to Branch and Bound.

In a field as competitive and diverse as Unit Commitment, comparisons with a variety of successful methods should be presented. Much research demonstrated improvements over previous attempts of the same class of algorithm (i.e. a new Genetic Algorithm over older Genetic Algorithms), but failed to consider other approaches. Branch and Bound had demonstrated strong performance on large systems whilst modern research into Meta-Heuristics was ongoing. Test systems presented in much of that work (20-40 units in size) could have been tackled by Branch and Bound, and this omission could be a serious factor in the low impact the field has had on industry.

The reason this omission is so important is it represents a lack of awareness of industry practice and state-of-the-art in Unit Commitment. Failure to mention industry practices, or dismissal of those practices for incorrect reasons, will surely weaken the impact of the work in the eyes of industry practitioners considering new approaches. Chapter 3.5.2 proposed a repository of Unit Commitment model formulations to use as benchmark formulations representing the state-of-the-art in the field, analogous to IEEE test systems for various uses in Power Systems research. This would ensure all researchers were aware of advances in the field such as the convex hull research discussed in Chapter 2.3.

Going further, common algorithms for each class, Branch and Bound, Lagrangian Relaxation, Genetic Algorithms etc. would easily allow researchers to compare their proposals with a wide variety of existing, high performance, approaches. Clearer comparisons on recognised benchmarks facilitates strong development of a field and clearer analysis of promising directions for future research, as demonstrated by the success of the International Planning Competition (discussed in Chapter 4.1.3). Such

resources would perhaps provide a platform from which Meta-Heuristic algorithms can develop and gain industry uptake.

As Automated Planning has not been attempted for Unit Commitment, this work could also encounter similar problems to the Meta-Heuristics work discussed above. In taking this work forward, the surveys of Classical Methods (Chapter 2.2), Industry Practices (Chapter 2.3), and Meta-Heuristics (Chapter 3) should be used to form a representative collection of algorithms on which to compare performance. The model proposed in Chapter 5 considered many key aspects of the Unit Commitment problem formulation, and where omissions occurred (such as reserve requirements and inaccurate cost models) they were highlighted and possible resolutions discussed. The final model proposed in Chapter 6 utilises a dedicated sub-solver. This could be extended to consider very complex model formulations including the early oversights and other features such as network constraints. This approach could therefore be very flexible. Whilst tangible demonstrations of industry applications have not been shown, steps to indicate why they may be achievable have been taken. In this way it is hoped some of the lessons learnt from the critical review of Meta-Heuristics have been taken into account.

7.2.2 Unit Commitment as Planning: Lessons Learnt

Chapter 4.3 discussed in detail potential benefits to the planning community of developing an Automated Planning approach for Unit Commitment.

The analysis of the International Planning Competitions and the competing systems revealed that whilst planning for numeric optimisation in a temporal framework has been tackled to some extent, there has been a shift in focus. In some real-world temporal domains, such as the Airport, Rovers, and Satellite Domain, the competing planners could find optimal solutions and had excellent problem coverage, even when tackling domains with complex constructs such as Timed Initial Literals. The quantity and complexity of numeric domain variants lessened throughout competition iterations, implying a move towards temporal complexities such as concurrent reasoning and planning under uncertainty.

The community has previously displayed an interest in Power Systems Engi-

neering problems. IPC benchmark domains including the PSR problem and work such as [9, 14] and the inclusion of a summary of the work presented here at ICAPS 2013 [192] all demonstrate a desire of the planning community to tackle real-world engineering problems. Justification for researching a planning approach to Unit Commitment can be drawn from the work on battery schedules [188] which contains many similarities to the Unit Commitment problem; an exogenous load is to be served with a limited resource. Key complexities of Unit Commitment over that work include:

- The concurrent scheduling of multiple sources of resource, increasing the combinatorial complexity.
- The ability to vary the amount of resource provided, increasing the numerical complexity.
- More complex cost functions.
- A fixed timeline and numeric goal (to minimise cost over a given period), rather than maximising total battery life, which is purely temporal.

These complexities represent a significant increase in difficulty over the battery problem and so a successful implementation would represent a contribution to the planning community.

The IPC praised planners with high problem coverage and general applicability. With this in mind a planning model in which the entire problem was modelled was developed in Chapter 5. Modelling difficulties arose, but it appeared the model was too complex for a planner to tackle unmodified. This lead to the development of the Separated Model presented in Chapter 6. Whilst this goes against the idea of general applicability praised in the IPC, it mirrors the development of the solver for the Battery problem. There a domain specific solver was developed before the authors generalised the approach to a specific class of problems. Supposing the extensions presented in Chapter 6.4 are successful, it is possible a more general solver can be developed to tackle all problems of a similar structure, i.e. numeric optimisation over a fixed horizon. This would represent a significant contribution to

the planning community, allowing planning to be a strong candidate for those early temporal-numeric domains not tackled for some time.

Chapter 5.1 then presented a detailed discussion of why Automated Planning would be a suitable candidate for Unit Commitment from an engineering viewpoint. The following ideas were highlighted:

- The finely grained temporal discretisation necessary for modelling demand contrasts with the much longer time frames a generating unit must typically remain online or offline for.
- The strength of planning in temporally complex problems demonstrated throughout the literature review in Chapter 4.2.
- The object-action formulation is a more natural fit, with less online / offline switch actions that MIP binary variables.
- Planning under uncertainty is receiving a lot of attention and the may provide efficient alternatives to costly Stochastic Unit Commitment processes, but before they can be applied a solid foundation tackling deterministic cases should be sought.

A model was developed to begin initial testing. Due to certain constraints the only planner available to tackle the model was UPMurphi. This highlights the lack of planners able to tackle such a problem, and the contribution successfully handling this domain would represent. Full details are discussed in Chapter 5.3. The key lessons learnt from attempting to solve this model were as follows:

- The complexity of modelling the costs requires more complex planners, of which there are few, and the complex language constructs required (Timed Initial Literals, Timed Initial Fluents and '`#t`' Continuous Processes) inherently extend the solution time.
- Enumerating all possible time points for switching actions and ramping actions is huge and not beneficial. Developing a system to efficiently find the correct subset (e.g. dynamic discretisation) appears instructive on small systems but

becomes too complex on large systems where the discretisation becomes very fine.

- Myopic optimisation is unsuited for Unit Commitment where the full operational cost for a unit's online duration should be taken into account, not just how much it costs to start up. An informative heuristic, whose use remains efficient when the frequency of its calculation is as high as it is when ramping actions are being considered, could not be found.

The above points combine to negate the perceived benefit of a planning approach. The removal of a fine time granularity and a reduced number of action points to consider in search are both lost in this model. Thus a second model which retained the planning advantages was developed in Chapter 6. The actions tracking individual unit output and cost were removed from the model and a simple numeric constraint was imposed over the entire planning horizon. This formulation has the following benefits over the initial model:

- Most of the actions from the original planning model are removed, reducing the state space (as the number of applicable actions is greatly reduced over the initial model) reducing reasoning time at each step as applicable actions are typically tested in a breadth first search manner.

- From a planning perspective this is a temporal problem with numeric constraints (min / max generation levels within bounds of Timed Initial Fluents). Any optimisation comes curtesy of the heuristic and external solver making this a domain typical of many in the IPC iterations.

- The custom heuristic (presented in Chapter 6.4), intended to test state validity and provide guidance on plan quality, can utilise existing work. The method presented is a simple algorithm but can be extended to use existing efficient Priority List methods. State validity can be incorporated utilising the vast body of work on linear LPs and Economic Dispatch. Reserves and network considerations can be included for varying levels of complexity and realism.

As the cost, the key feature for optimisation, was now removed from the planner's consideration, running tests on such a model would not be instructive. Instead a simple, domain specific heuristic was proposed. Initial work on implementing this approach using POPF revealed the following key points:

- Two separate approaches using different heuristics and search algorithms could be used.
  - An A* approach considering both cost so far and cost to go would intuitively provide a more accurate search at the cost of increased memory usage and solution time. Current barriers to implementation are grounding the temporal actions in the partial plan, communicating efficiently with an external solver such as CPLEX to avoid bottlenecks and efficient warm starting to quickly solve the LP.
  - An Enforced Hill Climbing approach would avoid the need to calculate the cost so far, removing the barriers to implementing the A* search. Attempting to implement this revealed the STN problem (see below).
- The STN, a key feature in the temporal strength of POPF, needed adaptations to correctly work with the heuristic returned. The exact timestamp of an action was not always resolved to correct end of its feasible interval, resulting in sub-optimal plans.
- The number of states expanded during search was greatly reduced over the original model.

This model represented a significant step forwards over the original and whilst time did not permit the completion of a planner able to tackle this model, significant steps were made. Chapter 6.5 details the above problems and possible solutions in further detail.

This thesis has presented two separate approaches for tackling Unit Commitment using Automated Planning. The motivations for each have been made clear and attempts to solve them have demonstrated that successfully handling this problem would push the boundaries of planning research. The complexity of the models

presented and the barriers encountered in attempting to solve them demonstrate that modelling Unit Commitment as a planning problem is itself non-trivial, and represents a contribution to the planning community. It is hoped that the problems encountered and discussions on possible resolutions can facilitate the development of and extended version of POPF to solve small test systems.

Rigorous testing against systems taken from the surveys presented should give a clear indication of whether Automated Planning is a viable approach in terms of solution quality and solution time. From here, more realistic models including reserves and larger test systems can be tackled. It is in these larger test cases where Branch and Bound has increasingly long solution times. Thus it is here where planning should demonstrate a reduction in solution time with comparable solution quality in order to be considered a viable option over Meta-Heuristics and alongside Branch and Bound.

Further work in this area may also reveal ways in which the techniques used to solve this problem can be abstracted. Unit Commitment is a numeric optimisation in a temporal domain with a fixed timeline and complex constraints. Successful abstraction of domain specific features of a Unit Commitment solution methodology would highlight new opportunities for novel approaches to other problems with these features. In this way separated models for temporal numeric problems using a hybrid Automated Planning and Mathematical Programming approach could be used for a wide class of problems, expanding the scope of Automated Planning as a field.

Finally utilising emerging research into planning under uncertainty could see an online planning approach to Unit Commitment with high levels of variable generation to rival Stochastic Unit Commitment, an area of critical importance to power systems of today. It is hoped that the research presented in this thesis has highlighted ideas, and can provide a starting point, from which both Automated Planning and Unit Commitment can benefit.

Appendix A

Unit Commitment Appendices

A.1 A Typical MIP Formulation

A MIP formulation solved using Branch and Bound is perhaps the most common approach to Unit Commitment today. Aspects of the formulation have been referred to throughout this thesis and so it is beneficial to make a typical formulation explicit here.

Formulation

This is a large and complex optimisation problem. The formulation given includes the constraints on the convex hull of the minimum online / offline subproblem as detailed in [41], which was published in 2006 and so should be a standard model for comparison in all modern implementations. Newer work [42, 43] tightens the formulation further but given its publication in 2012 and 2013 is less well known than [41] and has not been included here for discussion. The model in [43] represents the state-of-the-art in MIP models and is therefore the one which should be used for detailed model comparisons, however adds little illustrative benefit to an introductory description such as this.

- n generating units (indexed below with u for clarification), each with the following characteristics
  - A fixed start up time $t_{u,\text{start}}$. Having been switched off, this is the amount

of time it takes for the generating unit u to come online and output power. During this time no power is output, and after this time the output is the minimum stable generation level (see below).

  - A fixed switch off time $t_{u,\text{off}}$. Having been switched on, this is the amount of time it takes for the generating unit u to come offline. The generating unit u must be outputting its minimum stable generation level (see below) before it can be switched off, and once switching off begins no power is output.
  - A fixed minimum run and minimum off time, $t_{u,\text{min on}}, t_{u,\text{min off}}$. Once a unit is switched on (off) it must remain on (off) for at least $t_{u,\text{min on}}$ ($t_{u,\text{min off}}$).
  - A minimum stable generation level $G_{\text{min},u}$. The generating unit u cannot output less power than this.
  - A maximum stable generation level $G_{\text{max},u}$. The generating unit u cannot output more power than this.
  - The maximum increase (decrease) in output of a generating unit u is equal to $R_{u,+(-)}$ MW / min. This is known as the ramp rate.
  - A start up cost $C_{u,\text{start}}$. The cost of switching on a generating unit u.
  - A no-load cost $C_{u,\text{no-load}}$. The cost of having the generating unit u online regardless of output.
  - A marginal running cost $C_{u,\text{marginal}}$. The per MW cost of output from the generating unit u.

- The portfolio of generating units must serve a deterministic demand over a period of length T. The output from all generating units must be greater than the demand at all times.
- The demand should be served at the lowest cost.
- Any combination of generating units is permitted so long as the total supply is greater than the demand.

- A cold start (with all units off) is not assumed. Instead, the system will have an initial configuration given by the end state of the previous day. Switching a unit on or off near the end of the day will mean it may not serve the full $t_{u,\text{min on}}$ or $t_{u,\text{min off}}$ in that same day. This creates starting parameters $t_{u,\text{initial on}}$ and $t_{u,\text{initial off}}$ which will be fixed for a given problem instance. These parameters can also define must run generation by setting $t_{u,\text{initial on}} = T$.

Indices

- $\mathcal{U} := \{u\}$: The set of generating units, to be indexed by u
- $i = 1 \dots m$: Indexes the time periods over which generation and demand are assumed constant.
- t_i : points in continuous time such that $t_0 = 0, t_m = T$ and there are m distinct time intervals $[t_{i-1}, t_i)$.

Parameters

- $\Delta t = T/m$: The time granularity, such that $t_i + \Delta t = t_{i+1}$ and $t_i = i \cdot \Delta t$.
- $G_{\min}(u)$: the minimum stable generation level of the generating unit u
- $G_{\max}(u)$: the maximum stable generation level of the generating unit u
- $R_{u,+(-)}$: the maximum ramp rate of the generating unit u. $R_{u,\pm} \geq 0$
- $T_{\text{switch on}}(u),\ T_{\text{switch off}}(u) \in \{0, \dots, m\}$: fixed times to switch a generating unit u on or off, these are in number of time periods to remove the need for floor or ceiling functions within the constraints.
- $T_{\text{min on}}(u),\ T_{\text{min off}}(u)$: minimum running times and off times for the generating unit u
- $T_{\text{total off}} := T_{\text{switch on}}(u) + T_{\text{min off}}(u) + T_{\text{switch off}}(u)$: defining a total off period reduces the number of binary decision variables needed and simplifies the formulation of the constraints. See [116] who first introduced this idea.

- $T_{\text{initial on}}(u), T_{\text{initial off}}(u)$: from the previous day a unit may still be required to be left on or off for a while.
- $C_{u,\text{start}}$, $C_{u,\text{no-load}}$, $C_{u,\text{marginal}}$: the costs as defined above for the generating unit u
- $D(i), i = 1, \ldots, m$: the demand for period i

Decision Variables

- $o(u,i) \in \{0,1\}$: binary flag for whether or not a generating unit is on during period i
- $y(u,i) \in \{0,1\}$: binary flag indicating the unit has turned on at period i. This is used to tighten the feasible region of the search space as demonstrated in [41]. $o(u,i) - o(u,i-1) = 1 \implies y(u,i) = 1$
- $z(u,i) \in \{0,1\}$: binary flag indicating the unit has turned off at period i. This is used in conjunction with $y(u,i)$ to tighten the feasible region of the search space as demonstrated in [41]. $o(u,i) - o(u,i-1) = -1 \implies z(u,i) = 1$
- $g(u,i) \in \{0\} \cup [G_{\min}(u), G_{\max}(u)]$: the exact output of the generating unit u during period i
- $c(u,i)$: cost incurred by bringing the generating unit u online for period i, i.e. at time t_{i-1}. The need for this is brought about by the simplification of $T_{\text{total off}}(u)$. See the start up cost constraint section for clarification.

Objective

Minimise the total cost:

$$\text{minimise} \sum_{i=1}^{m} \sum_{u \in \mathcal{U}} \Big[g(u,i) \cdot C_{\text{marginal}}(u) \cdot \Delta t + o(u,i) \cdot C_{\text{no load}}(u) \cdot \Delta t + c(u,i) \Big] \quad \text{(A.1.1)}$$

Constraints

Balanced:

$$\sum_{u \in \mathcal{U}} g(u,i) >= D(i) \qquad \forall\ i \tag{A.1.2}$$

Generation Limits:

$$\begin{aligned} g(u,i) &\geq o(u,i) \cdot G_{\min}(u) \\ g(u,i) &\leq o(u,i) \cdot G_{\max}(u) \end{aligned} \qquad \forall\ i, \quad \forall\ u \tag{A.1.3}$$

Ramp rates:

$$\begin{aligned} g(u,i+1) &\leq g(u,i) + \Delta t \cdot R_{u,+} + (G_{\max}(u) - \Delta t \cdot R_{u,+}) y(u,i+1) \\ g(u,i+1) &\geq g(u,i) - \Delta t \cdot R_{u,-} - (G_{\max}(u) - \Delta t \cdot R_{u,-}) z(u,i+1) \end{aligned} \qquad \forall\ i < m, \quad \forall\ u \tag{A.1.4}$$

Start Up Cost:

$$\begin{aligned} c(u,i) &\geq y(u,i) \cdot C_{u,\text{start}} && \forall\ i, \quad \forall\ u \\ c(u,i) &\geq 0 && \forall\ i, \quad \forall\ u \end{aligned} \tag{A.1.5}$$

Minimum on time:

$$\sum_{j=1}^{T_{\text{initial on}}(u)} o(u,j) = T_{\text{initial on}}(u) \qquad \forall u \in \mathcal{U} \tag{A.1.6}$$

$$\sum_{j=n}^{n+T_{\min\ \text{on}}-1} o(u,j) \geq T_{\min\ \text{on}}(u)\ y(u,n) \qquad \forall\ u \in \mathcal{U} \quad \forall\ n \in \mathcal{T}_{\text{on mid}}(u) \tag{A.1.7}$$

$$\sum_{j=n}^{m} [o(u,j) - y(u,n)] \geq 0 \qquad \forall\ u \in \mathcal{U} \quad \forall\ n \in \mathcal{T}_{\text{on end}}(u) \tag{A.1.8}$$

where

$$\mathcal{T}_{\text{on mid}}(u) = \{i | i \in \{T_{\text{initial on}}(u) + 1, \ldots, m - T_{\min\ \text{on}}(u)\}\} \tag{A.1.9}$$

$$\mathcal{T}_{\text{on end}}(u) = \{i | i \in \{m + 1 - T_{\min\ \text{on}}(u), \ldots, m\}\} \tag{A.1.10}$$

As the total-off variable has been introduced the switch off time, minimum off

time and switch on time are all dealt with in the constraints:

$$\sum_{j=1}^{T_{\text{initial off}}(u)} o(u,j) = 0 \qquad \forall\ u \in \mathcal{U} \tag{A.1.11}$$

$$\sum_{j=n}^{n+T_{\text{total off}}-1} [1 - o(u,j)] \geq T_{\text{total off}}(u)\ z(u,n) \qquad \forall\ u \in \mathcal{U} \quad \forall\ n \in \mathcal{T}_{\text{off mid}}(u) \tag{A.1.12}$$

$$\sum_{j=n}^{m} [1 - o(u,j) - z(u,n)] \geq 0 \qquad \forall\ u \in \mathcal{U} \quad \forall\ n \in \mathcal{T}_{\text{off end}}(u) \tag{A.1.13}$$

where

$$\mathcal{T}_{\text{off mid}}(u) = \{i | i \in \{T_{\text{initial off}}(u) + 1, \ldots, m - T_{\text{min on}}(u)\}\} \tag{A.1.14}$$

$$\mathcal{T}_{\text{off end}}(u) = \{i | i \in \{m + 1 - T_{\text{min off}}(u), \ldots, m\}\} \tag{A.1.15}$$

Tight Constraints added as in [41] :

$$o(u,i-1) - o(u,i) + y(u,i) - z(u,i) = 0 \qquad \forall\ i > 1 \quad \forall\ u \in \mathcal{U} \tag{A.1.16}$$

$$\sum_{j=n-T_{\text{min on}}(u)+1}^{n} y(u,j) \leq o(u,n) \qquad \forall\ n \in [\max(T_{\text{initial on}}(u) + 1, T_{\text{min on}}(u)), m] \quad \forall\ u \in \mathcal{U} \tag{A.1.17}$$

$$o(u,n) + \sum_{j=n-T_{\text{total off}}(u)+1}^{n} z(u,j) \leq 1 \qquad \forall\ n \in [\max(T_{\text{initial off}}(u) + 1, T_{\text{total off}}(u)), m] \quad \forall\ u \in \mathcal{U} \tag{A.1.18}$$

A.1.1 Summary of Results

For illustrative purposes a series of tests were ran on a desktop PC. 12 demand profiles from 2009 UK demand data were selected at regular intervals throughout the year to give a small sample possible demand profiles. They are based on the half hourly demand data for GB available from the National Grid website [193]. The data used is the 1st of each month of 2009. During 2009 the peak load was 58,554 MW and the min load was 19,556. The data was scaled such that the peak demand was just satisfiable in all cases but the systems would be stretched. In smaller systems the problems are trivial if this is not the case, typically being solvable using 1 unit.

5 generation portfolios were drawn up with 6, 12, 17, 34 and 51 units. The 6 unit system data was from taken [58] and the 12 unit data included 6 additional units with properties similar to those in the 6 unit system. The 17, 34 and 51 unit systems were 1/3, 2/3 and all of the units from the Irish All Island Grid Project, the data for which is publicly available [194].

The above model was implemented in the commercially available AIMMS package, which calls the well known CPLEX solver. CPLEX was able to solve this problem to optimality in a number of cases, Table A.1 gives details of these problems and solutions. For portfolios of size 34 and above optimal solutions were not found. These results are given in A.2.

These results demonstrate that the method is not only feasible on single workstations but also competitive. All solution bar 1 is found in under an hour, and the ability to run the solver in parallel to easily speed up the solution process, as occurs in industry, should clearly indicate to all researchers in the field that a MIP implementation should be compared against in any research.

ID	Unit Count (Capacity) (MW)	Peak Demand (%)	Vars	(Ints)	Constrs	Solution Time	
1	6 (300)	48	1441	(864)	2634	0.109	
2		52				0.109	
3		49				0.109	
4		50				0.094	
5		44				0.187	
6		39				0.172	
7		50				0.125	
8		35				0.125	
9		45				0.109	
10		51				0.093	
11		46				0.188	
12		59				0.172	
13	12 (600)	48	2881	(1728)	5219	1.014	
14		52				0.421	
15		49				0.499	
16		50				0.234	
17		44				0.968	
18		39				0.327	
19		50				0.281	
20		35				2.091	
21		45				0.406	
22		51				0.374	
23		46				1.342	
24		59				0.421	
25	17 (2,726)	48	4081	(2448)	7333	51.558	(00:00:52)
26		52				54.819	(00:00:55)
27		49				30.124	(00:00:30)
28		50				5.008	(00:0:05)
29		44				4.462	(00:00:04)
30		39				1.155	(00:00:01)
31		50				30.654	(00:00:31)
32		35				1.966	(00:00:02)
33		45				1.826	(00:00:02)
34		51				13.931	(00:00:13)
35		46				7.722	(00:00:08)
36		59				114.442	(00:02:54)

Table A.1: Table summarising the problem statistics for a series of demand profiles on generating portfolios of increasing size. There were 12 distinct demand profiles taken from UK data for 2009. These were scaled down from an assumed initial generating capacity of 60 GW to a system of 300 MW. The profiles were then scaled up by the amount of generating capacity increase. These problems were all reported as being solved to optimality.

ID	Unit Count (Capacity) (MW)	Peak Demand (%)	Vars	(Ints)	Constrs	Solution Time	Gap (%)
37	34 (5,286)	48	8,161	(4,896)	14,568	2,942	(00:49:03)	1.094
38		52				2,177.306	(00:36:17))	1.106
39		49				3,763.914	(01:02:44)	0.476
40		50				2,833.946	(00:47:14)	0
41		44				2,075.734	(00:34:36)	1.255
42		39				4,089.411	(01:08:09)	0
43		50				1,720.457	(00:28:40)	0.802
44		35				2,897.952	(00:48:18)	0
45		45				5,837.588	(01:37:18)	0
46		51				7,359.16	(02:02:39)	0
47		46				3,509.383	(00:28:29)	0.707
48		59				2,784.946	(00:46:25)	0.768
49	51 (8,488)	48	12,241	(7,344)	21,827	1,851.279	(00:30:51)	1.395
50		52				1,871.372	(00:31:11)	1.104
51		49				2,261.671	(00:37:42)	0.911
52		50				2,835.303	(00:47:15)	0.214
53		44				14,995.628	(04:09:56)	0.575
54		39				2,201.315	(00:36:41)	1.024
55		50				2,659.568	(00:44:20)	0.224
56		35				2,447.172	(00:40:47)	1.214
57		45				1,681.628	(00:28:02)	0.750
58		51				2,316.255	(00:38:36)	0.414
59		46				2,271.343	(00:37:51)	0.934
60		59				2,270.377	(00:37:50)	0.833

Table A.2: Table summarising the problem statistics for a series of demand profiles on generating portfolios of increasing size. There were 12 distinct demand used as in Table A.1. For problems with a non-zero gap, the gap was calculated as : $(z_{\text{best integer}} - z_{\text{linear}})/z_{\text{best integer}}$

Appendix B

Planning Appendices

B.1 UPMurphi Heuristic in More Detail

As discussed in Chapter 5.3 using the total cost plus some estimation of the future cost (which would never be either accurate or provably admissible and appreciably different for each state) proved ineffective. Instead a 'greedy' heuristic was tried. As each 'interesting' action is durative the heuristic is calculated before (and after) each durative action over the time period that durative action will (or did) cover. Let the **Section Cost** be the cost of operation over the past Δt minutes in state n, given by

$$S(n, \Delta t) := \sum_u \Big[C_{\text{no load}}(u)\Delta t + C_{\text{marginal}}(u)g(u)\Delta - \mathbb{I}_{\{\text{ramping up},u\}}(\Delta t \Delta t R_u^+/2) + \mathbb{I}_{\{\text{ramping down},u\}}(\Delta t \Delta t R_u^-/2) \Big]$$

where $g(u)$ is the output of the unit u at the end of the period Δt. Let the **Projected Section Cost** be the cost of operation over the coming Δt minutes in state n, given by

$$\tilde{S}(n, \Delta t) := \sum_u \Big[C_{\text{no load}}(u)\Delta t + C_{\text{marginal}}(u)g(u)\Delta t + \mathbb{I}_{\{\text{ramping up},u\}}(\Delta t \Delta t R_u^+/2) - \mathbb{I}_{\{\text{ramping down},u\}}(\Delta t \Delta t R_u^-/2) \Big]$$

where $g(u)$ is the output of the unit u at the start of the period Δt.

UPMurphi is set to prefer those states with higher heuristic values, so a simple option would be to set $T := -S(n, \Delta t)$ where Δt is the time just passed, or when a ramp has begun $T := -\tilde{S}(n, \Delta t)$ where Δt is the minimum time the unit will ramp for (see below). Clearly this is biased towards actions with shorter durations but the cost per minute, as given by

$$T := \begin{cases} -S(n, \Delta t)/\Delta t & \text{if time has passed} \\ -\tilde{S}(n, \Delta t)/\Delta t & \text{if projecting a cost} \end{cases} \tag{B.1.1}$$

made good first attempts at the heuristic. In both cases t is at the start of the period given by Δt, as this removes weighting bias when comparing passing time for a long period of time and passing time having ramped down (typically a smaller duration and so the weighted discount for the later plan will be more than the discount for the cost, not the desired behaviour). On small problem instances these gave good performance however in larger instances the search stalled. It did so as the demand, and therefore the cost per minute, was much greater later in the day than at the start of the day. This meant that once a certain point in the day was reached the search backtracked to earlier, less optimal plans.

Weighting the heuristic with some notion of time and the demand being served at that moment in time would prevent this unnecessary backtracking behaviour. The scale of time is closer to that of cost per minute than of total cost but is that is still dependent on the portfolio size. Weighting as a proportion of the total planning horizon is independent of portfolio size and proved effective.

As time progresses through a typical planning horizon the demand increases so the cost per minute will also increase. The time weighting alone does not compensate for the fact that a later optimal plan serving a higher load will have a higher cost per minute than an earlier suboptimal plan serving a lower load. Defining the heuristic as a weighted cost per minute given the demand removes this bias towards times with lower demand whilst the time weighting continues to prevent the planner from

stalling in the search. The following heuristic proved effective.

$$T := \begin{cases} -\frac{S(n,\Delta t)}{\Delta t D(t)} \cdot \frac{H-t}{H} & \text{if time has passed} \\ -\frac{\tilde{S}(n,\Delta t)}{\Delta t D(t)} \cdot \frac{H-t}{H} & \text{if projecting a cost} \end{cases} \tag{B.1.2}$$

B.2 Full Unit Commitment Model : Pddl

This is the domain expressed in PDDL. As explain in the text this was converted to a UPMURPHI model as at time of development POPF was unable to solve this problem due to the non-linear cost of ramping and the Timed Initial Fluents coupled with the overall envelope constraint.

B.2.1 Domain

```
(define (domain UnitCommitment)
    (:requirements :typing :fluents :durative-actions :timed-initial-literals
        :negative-preconditions :duration-inequalities)
    (:types unit)
    (:predicates
     (on ?u - unit)
     (off ?u - unit)
     (canSwitchOn ?u - unit)
     (canSwitchOff ?u - unit)
     (complete)
     (demandServiced)
     (precondition-to-start)

     (rampingUp ?u - unit)
     (rampingDown ?u - unit))

    (:functions
     (supply)
     (demand)
     (totalCost)

     ;; unit parameters
     (output ?u - unit)
     (generationMin ?u - unit)
     (generationMax ?u - unit)
     (costStartUp ?u - unit)
     (costNoLoad ?u - unit)
     (costMarginal ?u - unit)
     (rampRateUp ?u - unit)
```

```
 (rampRateDown ?u - unit)
 (minimumOnTime ?u - unit)
 (totalOffTime ?u - unit))

(:durative-action switch_on
 :parameters (?u - unit)
 :duration (= ?duration (minimumOnTime ?u))
 :condition ( and
    (at start (off ?u))
    (at start (canSwitchOn ?u)))
 :effect ( and
    (at start (on ?u))
    (at start (not (off ?u)))
    (at start (not (canSwitchOn ?u)))
    (at end (canSwitchOff ?u))
    (at start (assign (output ?u) (generationMin ?u)))
    (at start (increase totalCost (costStartUp ?u)))))

(:durative-action switch_off
 :parameters (?u - unit)
 :duration (= ?duration (totalOffTime ?u))
 :condition ( and
    (at start (on ?u))
    (at start (canSwitchOff ?u)))
 :effect ( and
    (at start (not (on ?u)))
    (at start (off ?u))
    (at start (not (canSwitchOff ?u)))
    (at start (assign (output ?u) (generationMin ?u)))
    (at end (canSwitchOn ?u))))

(:durative-action envelope
 :parameters ()
 :duration (= ?duration 1440)
 :condition (and
    (over all (> supply demand))
    (at end (complete))
    (at start (precondition-to-start)))
 :effect (and
    (at end (demandServiced))))

;; To be ran for the times when the unit is generating a flat amount
(:durative-action generate
 :parameters (?u - unit)
 :duration (<= ?duration 1440)
 :condition ( and
```

```
      (over all (on ?u))
      (over all (not (rampingUp ?u)))
      (over all (not (rampingDown ?u))))
   :effect ( and
      (at start (increase (totalCost) (* ?duration
          (+ ( * (output ?u) (costMarginal ?u))  (costNoLoad ?u)  ))))
      (at start (increase (supply) (output ?u)))
      (at end (decrease (supply) (output ?u)))))

  (:durative-action ramp_up
   :parameters (?u - unit)
   :duration (<= ?duration
        (/ (- (generationMax ?u) (output ?u)) (rampRateUp ?u)))
   :condition ( and
      (over all (on ?u))
      (over all (not (rampingDown ?u)))
      (over all (<= (output ?u) (generationMax ?u))))
   :effect ( and
      (at start (rampingUp ?u))
      (at start (increase (supply) (output ?u)))
      (at start (increase (totalCost) (* ?duration
          (+ ( * (output ?u) (costMarginal ?u))  (costNoLoad ?u)  ))))
      (increase (output ?u) (* #t (rampRateUp ?u)))
      (increase (supply) (* #t (rampRateUp ?u)))
      (at end (not (rampingUp ?u)))
      (at end (decrease (supply) (output ?u)))))

  (:durative-action ramp_down
   :parameters (?u - unit)
   :duration (<= ?duration
       (/ (- (output ?u) (generationMin ?u)) (rampRateDown ?u)))
   :condition ( and
       (over all (on ?u))
       (over all (not (rampingUp ?u)))
       (over all (>= (output ?u) (generationMin ?u))))
   :effect ( and
       (at start (rampingDown ?u))
       (at start (increase (totalCost) (* ?duration
           (+ ( * (output ?u) (costMarginal ?u))  (costNoLoad ?u)  ))))
       (at start (increase (supply) (output ?u)))
       (decrease (output ?u) (* #t (rampRateDown ?u)))
       (decrease (supply) (* #t (rampRateDown ?u)))
       (at end (not (rampingDown ?u)))
       (at end (decrease (supply) (output ?u))))))
```

B.2.2 Problem

Below is a typical problem file. All units and their characteristics must be detailed here. The non-unit-specific variables `supply` and `totalCost` must be initialised to the correct values as the model only updates them.

```
(define (problem ucp_no_ramp_test)
    (:domain UnitCommitment)
    (:objects u0 ... - unit)
    (:init
    (off u0)
    ...
    (canSwitchOn u0)
    ....
    (= (output u0) 0)
    (= (generationMin u0) 60)
    (= (generationMax u0) 150)
    (= (costStartUp u0) 20000)
    (= (costNoLoad u0) 500)
    (= (costMarginal u0) 20)
    (= (rampRateUp u0) 2)
    (= (rampRateDown u0) 2)

    (= (minimumOnTime u0) 100)
    (= (totalOffTime u0) 100)
    ...
    (= (supply) 0)
    (= (totalCost) 0)

    (precondition-to-start)
    (at 0.005 (not (precondition-to-start)))
    (at 1440 (complete))

    ;; Init Timed Initial Fluents for demand
    (at 0 (= (demand) 1024))
    ...
    (:goal (and (complete) (demandServiced)))
    (:metric minimize (totalCost)))
```

B.3 Separated Unit Commitment Model : PDDL

B.3.1 Domain

```
(define (domain UnitCommitment)
```

```
(:requirements :typing :fluents :durative-actions
        :timed-initial-literals :negative-preconditions)
(:types unit)
(:predicates
     (on ?u - unit)
     (off ?u - unit)
     (canSwitchOn ?u - unit)
     (canSwitchOff ?u - unit)
     (complete)
     (demandServiced)
     (precondition-to-start))
(:functions
     (minimumOnTime ?u - unit)
     (totalOffTime ?u - unit)
     (maxSupply)
     (minSupply)
     (demand)
     (totalCost)

    (generationMin ?u - unit)
    (generationMax ?u - unit)
    (costStartUp ?u - unit)
    (costNoLoad ?u - unit)
    (costMarginal ?u - unit)
    (rampRateUp ?u - unit)
    (rampRateDown ?u - unit))

(:durative-action switch_on
 :parameters (?u - unit)
 :duration (= ?duration (minimumOnTime ?u))
 :condition ( and
         (at start (off ?u))
         (at start (canSwitchOn ?u)))
 :effect ( and
         (at start (on ?u))
         (at start (not (off ?u)))
         (at start (not (canSwitchOn ?u)))
         (at end (canSwitchOff ?u))
         (at start (increase maxSupply (generationMax ?u)))
         (at start (increase minSupply (generationMin ?u)))))

(:durative-action switch_off
 :parameters (?u - unit)
 :duration (= ?duration (totalOffTime ?u))
 :condition ( and
         (at start (on ?u))
```

```
            (at start (canSwitchOff ?u)))
     :effect ( and
            (at start (not (on ?u)))
            (at start (off ?u))
            (at start (not (canSwitchOff ?u)))
            (at end (canSwitchOn ?u))
            (at start (decrease maxSupply (generationMax ?u)))
            (at start (decrease minSupply (generationMin ?u)))))

    (:action dummy_cost
     :parameters ()
     :precondition ()
     :effect (and (increase totalCost 1)))

    (:durative-action envelope
     :parameters ()
     :duration (= ?duration 1440)
     :condition (and
            (over all (> maxSupply demand))
            (at end (complete))
            (at start (precondition-to-start)))
     :effect (and
            (at end (demandServiced))))
  )
```

B.3.2 Problem

```
(define (problem ucp_no_ramp_test)
    (:domain UnitCommitment)
    (:objects u0 u1 u2 - unit)
    (:init
    (off u0)
    ...
    (on u1)
    ...
    (canSwitchOn u0)
    ...
    (canSwitchOff u1)
    ...
     (= (maxSupply) 100)
     (= (minSupply) 50)

     (= (generationMin u0) 60)
     (= (generationMax u0) 150)
     (= (costStartUp u0) 20000)
     (= (costNoLoad u0) 500)
     (= (costMarginal u0) 20)
```

```
    (= (rampRateUp u0) 2)
    (= (rampRateDown u0) 2)

    (= (minimumOnTime u0) 100)
    (= (totalOffTime u0) 100)
    ...
    (precondition-to-start)
    (at 0.005 (not (precondition-to-start)))
    (at 1440 (complete))

    ;; Init Timed Initial Fluents for demand
    (at 0 (= (demand) 1024))
    ...
    )
   (:goal (and (complete) (demandServiced)))
   (:metric minimize (totalCost))
)
```

B.4 Priority List Algorithm

Below, an extension of the Priority List Algorithm detailed in Chapter 6.4.1 which includes the ability to switch the units off and optionally obey ramp rate constraints is presented. The differences are in the manipulations to the sets used and the complexity of the functions setting the outputs. Once implemented it is hoped this algorithm will provide more accurate costs but have similar solution times.

1. Split the set of generating units into two sets, $\mathcal{G}, \mathcal{C}$, for those currently online (**G**enervating) and those currently offline (**C**ool). Order these sets by marginal cost such that $\mathcal{G}_0$ is the lowest marginal cost of the units in $\mathcal{G}$ and $\mathcal{C}_0$ is the lowest marginal cost in $\mathcal{C}$.

2. Let

   - $D(i)$ be the demand for period i
   - $g(u, i)$ be the output from each generating unit
   - $S(i) = \sum_{u \in \mathcal{G}} g(u, i)$ be the total output
   - $\delta_i = S(i) - D(i)$ be the power deficit

- $R_+(u), R_-(u)$ be the maximum and minimum amounts by which a generating unit's output can ramp between time periods.

3. Let $i \leftarrow 1$ be the first period.

4. Set $g(u,i) \leftarrow G_{\min}(u)$ for all $u \in \mathcal{G}$. Update $S(i)$ and δ_i. Set $u = 0$.

5. *Assign outputs...*

   (a) *...without Ramp Rates:*

   $$g(u,i) \leftarrow \max\Big[\min(G_{\max}(u), \delta_i), G_{\min}(u)\Big]$$

   (b) *...with Ramp Rates:*

   $$g(u,i) \leftarrow \max\Big[\min\big[\min\big(G_{\max}(u), g(u,i-1) + R_+(u)\big), \delta_i\big], \max\big(G_{\min}(u), g(u,i-1) - R_-(u)\big)\Big]$$

   Update $S(i)$ and δ_i.

6. If $\delta_i \geq 0$

   (a) *Switch Off Units:*
   Set $u_M = \mathcal{G}_{|\mathcal{G}|}$. If $S(i) - g(u_M, i) > D(i)$

   $$g(u_M, i) = 0, \quad \mathcal{G} \leftarrow \mathcal{G} \backslash \{u_M\}, \quad \mathcal{C} \leftarrow \mathcal{C} \cup u_M, \quad \text{Go to Step 6a}$$

   (b) if $i = N$ { stop } else { $i \leftarrow i + 1$, Go to step 4 }

7. If $u < |\mathcal{G}| - 1$ { $u \leftarrow u + 1$, go to Step 5 }

8. *Switch On Additional Units:*

   (a) Remove $\mathcal{C}_0$ from $\mathcal{C}$ and insert into $\mathcal{G}$. Let v' be its sorted index in $\mathcal{G}$.

   (b) Set $g(v', i) \leftarrow \min\Big[\max\big[G_{\min}(v'), \delta_i\big], G_{\max}(v')\Big]$. Update $S(i)$ and δ_i.

   (c) If $\delta_i \geq 0$ { if $i = N$ { stop } else { $i \leftarrow i + 1$, Go to Step 4 } } else { Go to step 8 }

Bibliography

[1] National Grid. Electricity ten year statement: Appendix 1, November 2012. Available online at: http://www.nationalgrid.com/uk/Electricity/ten-year-statement/current-elec-tys/. Accessed Oct 2013.

[2] Dan Streiffert, Russ Philbrick, and Andrew Ott. A mixed integer programming solution for market clearing and reliability analysis. In *IEEE Power Engineering Society General Meeting*, volume 3, pages 2724–2731, 2005.

[3] Allen J. Wood and Bruce F. Wollenberg. *Power Generation, Operation, and Control*. John Wiley & Sons, Inc., 2nd edition, 1996.

[4] Richard P. ONeill. Recent iso software enhancements and future software and modeling plans. Technical report, Federal Energy Regulatory Commission, November 2011. Available online at : http://www.ferc.gov/industries/electric/indus-act/rto/rto-iso-soft-2011.pdf. Accessed Aug 2013.

[5] Matthew Musto and Muhammad Marwali. Mixed integer programming NYISO proof of concept experience, June 2013. Presentation of ongoing work available online at http://www.ferc.gov/CalendarFiles/20130710155009-M1_Musto.pdf: . Accessed Aug 2013.

[6] George Dantzig and Mukund Thapa. *Linear Programming 1: Introduction*. Springer, February 1997.

[7] National Grid. Electricity ten year statement, November 2012. Available online at: http://www.nationalgrid.com/uk/Electricity/ten-year-statement/current-elec-tys/. Accessed Oct 2013.

[8] Duncan S. Callaway and Ian A. Hiskens. Achieving controllability of electric loads. *Proceedings of the IEEE*, 99(1):184–199, January 2011.

[9] 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 *23rd International Conference on Automated Planning and Scheduling*. AAAI Publications, 2013. This paper has been accepted for publication but has not been published at time of writing. Is available online at: www.aaai.org/ocs/index.php/ICAPS/ICAPS13/paper/view/6047. Accessed Aug 2013.

[10] Gerald Sheble and George Fahd. Unit commitment literature synopsis. *IEEE Transactions on Power Systems*, 9(1):128–135, February 1994.

[11] Subir Sen and DP Kothari. Optimal thermal generating unit commitment: a review. *International Journal of Electrical Power & Energy Systems*, 20(7):443–451, 1998.

[12] Padhy, Narayana, and Prasad. Unit commitment-a bibliographical survey. *IEEE Transactions on Power Systems*, 19(2):1196–1205, 2004.

[13] T Logenthiran and D Srinivasan. Formulation of unit commitment (uc) problems and analysis of available methodologies used for solving the problems. In *2010 IEEE International Conference on Sustainable Energy Technologies (ICSET)*, pages 1–6. IEEE, 2010.

[14] K. Bell, A. I. Coles, M. Fox, D. Long, and A. J. Smith. The application of planning to power substation voltage control. In *ICAPS Workshop on Scheduling and Planning Applications (SPARK)*, 2008.

[15] K. Bell, A. Coles, A. Coles, M. Fox, and D. Long. The role of AI Planning as a decision support tool in power substation management. *AI Communications*, 22(1), 2009.

[16] Website for the 2012 FERC June Technical Conference : http://www.ferc.gov/industries/electric/indus-act/market-planning/2012-conference.asp. Accessed Aug 2013.

[17] Website for the 2013 FERC June Technical Conference : http://www.ferc.gov/industries/electric/indus-act/market-planning/2013-conference.asp.

[18] C.J. Baldwin, K. M. Dale, and R. F. Dittrich. A study of economic shutdown of generating units in daily dispatch. *AIEE Transactions on Power Apparatus and Systems*, 78(4):1272–1284, December 1959.

[19] P. G. Lowery. Generating unit commitment by dynamic programming. *IEEE Transactions on Power Apparatus and Systems*, PAS-85(5):422–426, May 1966.

[20] K.D. Le, J.T. Day, B. L. Cooper, and E. W. Gibbons. A global optimization method for scheduling thermal generation, hydro generation, and economy purchases. *IEEE Transactions on Power Apparatus and Systems*, PAS-102(7):1986–1993, July 1983.

[21] P. P J Van den Bosch and G. Honderd. A solution of the unit commitment problem via decomposition and dynamic programming. *IEEE Transactions on Power Apparatus and Systems*, PAS-104(7):1684–1690, 1985.

[22] Walter L. Snyder, H.David Powell, and John C. Rayburn. Dynamic programming approach to unit commitment. *IEEE Transactions on Power Systems*, 2(2):339–348, May 1987.

[23] C. K. Pang and H. C. Chen. Optimal short-term thermal unit commitment. *IEEE Transactions on Power Apparatus and Systems*, 95(4):1336–1346, July 1976.

[24] W.J. Hobbs, G. Hermon, S. Warner, and G.B. Shelbe. An enhanced dynamic programming approach for unit commitment. *IEEE Transactions on Power Systems*, 3(3):1201–1205, August 1988.

[25] F.N. Lee. Short-term thermal unit commitment-a new method. *IEEE Transactions on Power Systems*, 3(2):421–428, May 1988.

[26] John A. Muckstadt and Sherri A. Koenig. An application of lagrangian relaxation to scheduling in power-generation systems. *Operations Research*, 25(3):387–403, 1977.

[27] Marshall L. Fisher. An applications oriented guide to lagrangian relaxation. *Interfaces*, 15(2):10–21, March 1985.

[28] Dimitri P. Bertsekas. *Nonlinear Programming*. Athena Scientific, 2 edition, September 1999.

[29] A. Merlin and P. Sandrin. A new method for unit commitment at electricite de france. *IEEE Transactions on Power Apparatus and Systems*, PAS-102(5):1218–1225, May 1983.

[30] Dimitri Bertsekas, G. Lauer, Jr. Sandell, N., and Thomas A. Posbergh. Optimal short-term scheduling of large-scale power systems. *IEEE Transactions on Automatic Control*, 28(1):1–11, January 1983.

[31] Fulin Zhuang and F.D. Galiana. Towards a more rigorous and practical unit commitment by lagrangian relaxation. *IEEE Transactions on Power Systems*, 3(2):763–773, May 1988.

[32] Jonathan F. Bard. Short-term scheduling of thermal-electric generators using lagrangian relaxation. *Operations Research*, 36(5):pp. 756–766, 1988.

[33] S. Virmani, Eugene C. Adrian, Karl Imhof, and Shishir Mukherjee. Implementation of a lagrangian relaxation based unit commitment problem. *IEEE Transactions on Power Systems*, 4(4):1373–1380, November 1989.

[34] Benjamin F. Hobbs, Michael H. Rothkopf, Richard P. ONeill, and Hung po Chao, editors. *The Next Generation of Electric Power Unit Commitment Models*. Springer, 2002.

[35] A. H. Land and A. G. Doig. An automatic method of solving discrete programming problems. *Econometrica*, 28(3):497–520, July 1960.

[36] Manfred Padberg and Giovanni Rinaldi. A branch-and-cut algorithm for the resolution of large-scale symmetric traveling salesman problems. *SIAM Review*, 33(1):60–100, March 1991.

[37] PJM's new generation-scheduling software to save customers estimated $56 million; unit commitment system uses first of its kind technique. [Press release] Available online at : http://www.prnewswire.com/news-releases/pjms-new-generation-scheduling-software-to-save-customers-estimated-56-million-unit-commitment-system-uses-first-of-its-kind-technique-75146262.html. Accessed Aug 2013.

[38] Hailong Hui, Chien-Ning Yu, and Sainath Moorty. Reliability unit commitment in the new ercot nodal electricity market. In *IEEE Power and Energy Society General Meeting*, pages 1 – 8, July 2009.

[39] Roshan Chhetri. Security constrained unit commitment considering multiple regions. Master's thesis, University of New Brunswick (Canada), 2006. Available online at : http://dspace.hil.unb.ca:8080/handle/1882/43722. Accessed Aug 2013.

[40] Robert E. Bixby, Mary Fenelon, Zonghao Gu, Ed Rothberg, and Roland Wunderling. MIP: Theory and practice - closing the gap. In *System Modelling and Optimization: Methods, Theory, and Applications*. Kluwer Academic Publishers, 2000.

[41] Deepak Rajan and Samer Takriti. Minimum up/down polytopes of the unit commitment problem with start-up costs. Technical report, IBM, 2005.

[42] James Ostrowski, Miguel F. Anjos, and Anthony Vannelli. Tight mixed integer linear programming formulations for the unit commitment problem. *IEEE Transactions on Power Systems*, 27(1):39 – 46, February 2012.

[43] Germán Morales-España, Jesus M. Latorre, and Andres Ramos. Tight and compact MILP formulation for the thermal unit commitment problem. Accepted for publication in IEEE Transactions on Power Systems, 2013.

[44] International Energy Agency. World Energy Outlook 2009, July 2012. [Online] : http://www.iea.org/textbase/nppdf/free/2009/weo2009.pdf.

[45] European Commission, July 2012. [Online] : http://ec.europa.eu/clima/policies/package/index_en.htm.

[46] RenewableUK, November 2012. [Online] : http://www.renewableuk.com/en/renewable-energy/wind-energy/uk-wind-energy-database/index.cfm.

[47] The Committee on Climate Change, July 2012. [Online] : http://www.theccc.org.uk/sectors/power/scenarios.

[48] Pöyry Energy Consulting. Impact of intermittency: How wind variability could change the shape of the British and Irish electricity markets, July 2009. [Online] : http://www.uwig.org/ImpactofIntermittency.pdf.

[49] M.D. Ilic. From hierarchical to open access electric power systems. *Proceedings of the IEEE*, 95(5):1060 –1084, May 2007.

[50] S. Takriti, J. Birge, and E. Long. A stochastic model for the unit commitment problem. *IEEE Transactions on Power Systems*, 11(3):1497–1508, August 1996.

[51] R. Barth, H. Brand, P. Meibom, and C. Weber. A stochastic unit-commitment model for the evaluation of the impacts of integration of large amounts of intermittent wind power. In *9th International Conference on Probabilistic Methods Applied to Power Systems*. PMAPS, June 2006.

[52] Juan M. Morales, Antonio J. Conejo, and Juan Perez-Ruiz. Short-term trading for a wind power producer. *IEEE Transactions On Power Systems*, 25(1):554–564, February 2010.

[53] Chris J. Dent, Janusz W. Bialek, and Benjamin F. Hobbs. Opportunity cost bidding by wind generators in forward markets: Analytical results. *IEEE Transactions On Power Systems*, 26(3):1600–1608, August 2011.

[54] E. Bitar, A. Giani, R. Rajagopal, D. Varagnolo, P. Khargonekar, K. Poolla, and P. Varaiya. Optimal contracts for wind power producers in electricity markets. In *49th IEEE Conference on Decision and Control*, December 2010.

[55] P. Meibom, R. Barth, H. Brand, and C. Weber. Wind power integration studies using a multi-stage stochastic electricity system model. In *Power Engineering Society General Meeting*. IEEE, June 2007.

[56] Aidan Tuohy, Peter Meibom, Eleanor Denny, and Mark OMalley. Unit commitment for systems with significant wind penetration. *IEEE Transactions on Power Systems*, 24(2):592–601, May 2009.

[57] Peter Meibom, Rüdiger Barth, Bernhard Hasche, Heike Brand, Christoph Weber, and Mark OMalley. Stochastic optimization model to study the operational impacts of high wind penetrations in Ireland. *IEEE Transactions On Power Systems*, 26(3):1367–1379, August 2011.

[58] Lannoye, Flynn, and O'Malley. Evaluation of power system flexibility. *IEEE Transactions on Power Systems*, 27(2):922–931, May 2011.

[59] E. Ela and B. Kirby. ERCOT event on february 26, 2008: Lessons learned. Technical report, National Renewable Energy Laboratory, July 2008.

[60] S. Lindenberg, B. Smith, K. ODell, and E. DeMeo. 20% wind energy by 2030: Increasing wind energys contribution to u.s. electricity supply, July 2008. Available online at : http://www.nrel.gov/docs/fy08osti/41869.pdf. Accessed Aug 2013.

[61] Scalable, parallel stochastic unit commitment for improved day ahead and reliability operations, June 2012. Presentation of ongoing work available online at : http://www.ferc.gov/CalendarFiles/20120626085534-Tuesday,%20Session%20TD2,%20Watson.pdf. Accessed Aug 2013.

[62] Stochastic unit commitment: Scalable computation and experimental results, June 2013. Presentation of ongoing work available online at : http://www.ferc.gov/EventCalendar/Files/20130710155720-M4_Watson.pdf. Accessed Aug 2013.

[63] Stochastic unit commitment: Stochastic process modeling for load and renewables, June 2013. Presentation of ongoing work available online at : http://www.ferc.gov/EventCalendar/Files/20130710155701-M4_Woodruff.pdf. Accessed Aug 2013.

[64] Dzung Phan, Ali Koc, and Jayant Kalagnanam. Algorithms for solving large-scale stochastic unit commitment and security-constrained economic dispatch problems, June 2013. Presentation of ongoing work available online at : http://www.ferc.gov/CalendarFiles/20120627085958-Wednesday,%20Session%20A,%20Phan.pdf. Accessed Aug 2013.

[65] Hugo Simao, Warren Powell, and Boris Defourny. Smart-iso: Modeling uncertainty in the electricity markets, June 2013. Presentation of ongoing work available online at : http://www.ferc.gov/EventCalendar/Files/20130710155826-T1A_Simao.pdf. Accessed Aug 2013.

[66] Alexander Sturt and Goran Strbac. Efficient stochastic scheduling for simulation of wind-integrated power systems. *IEEE Transactions on Power Systems*, 27(1):323–334, February 2012.

[67] Pravin Varaiya, Felix Wu, and Janusz Bialek. Smart operation of smart grid: Risk-limiting dispatch. *Proceedings of the IEEE*, 99(1):40–57, January 2011.

[68] Ram Rajagopal, Eilyan Bitar, Pravin Varaiya, and Felix Wu. Risk-limiting dispatch for integrating renewable power. Technical report, Berkeley, September 2011.

[69] Peter B. Luh, Yaowen Yu, Bingjie Zhang, Eugene Litvinov, Tongxin Zheng, Feng Zhao, Jinye Zhao, and Congcong Wang. Grid integration of intermittent

wind generation: A markovian approach. Accepted for publication in IEEE Transactions on Smart Grid, August 2013.

[70] J. Mur-Amada and A. A. Bayod-Rujula. Wind power variability model part 1 - foundations. In *9th International Conference on Electrical Power Quality and Utilisation*, October 2007.

[71] F. L. Alvarado and R. Rajaraman. Optimal unit commitment under uncertainty in electricity markets, June 2013. Presentation of ongoing work available online at : http://www.ferc.gov/CalendarFiles/20130710160901-T3A_Alvarado.pdf. Accessed Aug 2013.

[72] Christian Blum and Andrea Roli. Metaheuristics in combinatorial optimization: Overview and conceptual comparison. *ACM Computing Surveys (CSUR)*, 35(3):268–308, 2003.

[73] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. Optimization by simulated annealing. *Science*, 220:671–680, 1983.

[74] Vlado Cerny. Thermodynamical approach to the traveling salesman problem: An efficient simulation algorithm. *Journal of Optimization Theory and Applications*, 45:41–51, January 1985.

[75] F. Zhuang and F. Galiana. Unit commitment by simulated annealing. *IEEE Transactions on Power Systems*, 5(1):311–318, February 1990.

[76] V. Granville, M. Krivanek, and J.-P. Rasson. Simulated annealing: a proof of convergence. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 16(6):652–656, 1994.

[77] A.H. Mantawy, Y.L. Abdel-Magid, and S.Z. Selim. A simulated annealing algorithm for unit commitment. *Power Systems, IEEE Transactions on*, 13(1):197 –204, February 1998.

[78] T. Senjyu, A.Y. Saber, N. Urasaki, and T. Funabashi. Fast technique for unit commitment by absolute stochastic simulated annealing. In *Power Engineering Society General Meeting, 2005. IEEE*, volume 2, pages 1293 – 1296, June 2005.

[79] D. Simopoulos and S. Kavatza. Consideration of ramp rate constraints in unit commitment using simulated annealing. In *Power Tech, 2005 IEEE Russia*, pages 1 –7, June 2005.

[80] D.N. Simopoulos, S.D. Kavatza, and C.D. Vournas. Unit commitment by an enhanced simulated annealing algorithm. *Power Systems, IEEE Transactions on*, 21(1):68 – 76, February 2006.

[81] S. Soliman and A. Mantawy. *Modern Optimization Techniques with Applications in Electric Power Systems*. Springer, 2012.

[82] D.N. Simopoulos, S.D. Kavatza, and C.D. Vournas. Unit commitment by an enhanced simulated annealing algorithm. *Power Systems, IEEE Transactions on*, 21(1):68 – 76, February 2006.

[83] Fred Glover and Claude McMillan. The general employee scheduling problem: an integration of MS and AI. *Computers and Operations Research - Special Issue: Applications of integer programming*, 13:563 – 573, May 1986.

[84] Fred Glover. Tabu search - part 1. *ORSA Journal on Computing*, 1(3):190–206, 1989.

[85] Fred Glover. Tabu search - part 2. *INFORMS Journal on Computing*, 2(1):4–32, 1990.

[86] Hiroyuki Mori and Takayuki Usami. Application of tabu search to unit commitment in power systems. In *Proceedings of ICEE*, pages 240–243, July 1995.

[87] H. Mori and T. Usami. Unit commitment using tabu search with restricted neighborhood. In *International Conference on Intelligent Systems Applications to Power Systems*, pages 422–427, 1996.

[88] A. Mantawy, Y. Abdel-Magid, and S. Selim. Unit commitment by tabu search. In *Generation, Transmission and Distribution, IEE Proceedings*, volume 145, pages 56–64, January 1998.

[89] Hiroyuki Mori and Osamu Matsuzaki. Embedding the priority list into tabu search for unit commitment. In *IEEE Power Engineering Society Winter Meeting*, volume 3, pages 1067 – 1072, 2001.

[90] D. Dasgupta and D.R. McGregor. Thermal unit commitment using genetic algorithms. *Generation, Transmission and Distribution, IEE Proceedings-*, 141(5):459 –465, September 1994.

[91] Gerald Sheblé and Timothy Maifeld. Unit commitment by genetic algorithm and expert system. *Electric Power Systems Research*, 11(3):1359 –1370, August 1996.

[92] T.T. Maifeld and G.B. Sheblé. Genetic-based unit commitment algorithm. *IEEE Transactions on Power Systems*, 11(3):1359 –1370, August 1996.

[93] S.A. Kazarlis, A.G. Bakirtzis, and V. Petridis. A genetic algorithm solution to the unit commitment problem. *Power Systems, IEEE Transactions on*, 11(1):83 –92, February 1996.

[94] K. A. Juste, H. Kita, E. Tanaka, and J. Hasegawa. An evolutionary programming solution to the unit commitment problem. *IEEE Transactions on Power Systems*, 14(4):1452–1459, November 1999.

[95] I.G. Damousis, A.G. Bakirtzis, and P.S. Dokopoulos. A solution to the unit-commitment problem using integer-coded genetic algorithm. *IEEE Transactions on Power Systems*, 19(2):1165–1172, May 2004.

[96] T. Senjyu, H. Yamashiro, K. Uezato, and T. Funabashi. A unit commitment problem by using genetic algorithm based on unit characteristic classification. In *Power Engineering Society Winter Meeting, 2002. IEEE*, volume 1, pages 58 – 63 vol.1, 2002.

[97] T. Senjyu, H. Yamashiro, K. Shimabukuro, K. Uezato, and T. Funabashi. Fast solution technique for large-scale unit commitment problem using genetic algorithm. *Generation, Transmission and Distribution, IEE Proceedings-*, 150(6):753 – 760, November 2003.

[98] A. Rudolf and R. Bayrleithner. A genetic algorithm for solving the unit commitment problem of a hydro-thermal power system. *Power Systems, IEEE Transactions on*, 14(4):1460 –1468, November 1999.

[99] J. Kennedy and R. Eberhart. Particle swarm optimization. In *Neural Networks, 1995. Proceedings., IEEE International Conference on*, volume 4, pages 1942–1948, 1995.

[100] Zwe-Lee Gaing. Discrete particle swarm optimization algorithm for unit commitment. In *Power Engineering Society General Meeting, 2003, IEEE*, volume 1, pages 418–424, July 2003.

[101] T.O. Ting, M.V.C. Rao, and C.K. Loo. A novel approach for unit commitment problem via an effective hybrid particle swarm optimization. *IEEE Transactions on Power Systems*, 21(1):411 – 418, February 2006.

[102] T. Logenthiran and D. Srinivasan. Particle swarm optimization for unit commitment problem. In *Probabilistic Methods Applied to Power Systems (PMAPS), 2010 IEEE 11th International Conference on*, pages 642 –647, June 2010.

[103] V.S. Pappala and I. Erlich. A new approach for solving the unit commitment problem by adaptive particle swarm optimization. In *Power and Energy Society General Meeting - Conversion and Delivery of Electrical Energy in the 21st Century, 2008 IEEE*, pages 1–6, July 2008.

[104] V.S. Pappala and I. Erlich. A variable-dimension optimization approach to unit commitment problem. *Power Systems, IEEE Transactions on*, 25(3):1696 –1704, August 2010.

[105] R. Tyson and G. Heydt. An expert system and dynamic programming hybrid for unit commitment. In *Proceedings of the Twenty-First Annual Power Symposium*, pages 2–11, October 1989.

[106] Z. Ouyang and S. Shahidehpour. A hybrid artificial neural network-dynamic programming approach to unit commitment. *IEEE Transactions on Power Systems*, 7(1):236 – 242, February 1992.

[107] Shyh-Jier Huang and Ching-Lien Huang. Application of genetic-based neural networks to thermal unit commitment. *IEEE Transactions on Power Systems*, 12(2):654–660, May 1997.

[108] A. H. Mantawy, Y.L. Abdel-Magid, and S.Z. Selim. Integrating genetic algorithms, tabu search, and simulated annealing for the unit commitment problem. *IEEE Transactions on Power Systems*, 14(3):829–836, August 1999.

[109] Chuan-Ping Cheng, Chih-Wen Liu, and Chun-Chang Liu. Unit commitment by lagrangian relaxation and genetic algorithms. *IEEE Transactions on Power Systems*, 15(2):707–714, May 2000.

[110] Haoyong Chen and Xifan Wang. Cooperative coevolutionary algorithm for unit commitment. *IEEE Transactions on Power Systems*, 17(1):128–133, February 2002.

[111] J. M S Pinheiro, C. R R Dornellas, M. Th Schilling, A.C.G. Melo, and Mello JCO. Probing the new IEEE reliability test system (rts-96): Hl-ii assessment. *IEEE Transactions on Power Systems*, 13(1):171–176, 1998.

[112] Gwo-Ching Liao. Application of novel hybrid techniques for short-term unit commitment problem. In *Proceedings of the IEEE Asia-Pacific Conference on Circuits and Systems*, volume 2, pages 1169 – 1172, December 2004.

[113] P. Sriyanyong. Unit commitment using particle swarm optimization combined with lagrange relaxation. In *IEEE Power Engineering Society General Meeting*, volume 3, pages 2752 – 2759, June 2005.

[114] Lawrence Jenkins. Hybrid algorithms for power system unit commitment. In *50th Midwest Symposium on Circuits and Systems*, pages 678–681, August 2007.

[115] M. Bavafa. A new hybrid approach for unit commitment using lagrangian relaxation combined with evolutionary and quadratic programming. In *Asia-Pacific Power and Energy Engineering Conference*, pages 1–6, March 2009.

[116] Miguel Carrion and Jose Arroyo. A computationally efficient mixed-integer linear formulation for the thermal unit commitment problem. *Proceedings of the IEEE*, 21(3):1371–1378, August 2006.

[117] F. Leanez and R. Palma-Behnke. Towards more accurate unit commitment performance comparisons. In *Transmission Distribution Conference and Exposition: Latin America, 2006. TDC '06. IEEE/PES*, pages 1–8, August 2006.

[118] G. K. Purushothama and Lawrence Jenkins. Simulated annealing with local search-a hybrid algorithm for unit commitment. *IEEE Transactions on Power Systems*, 18(1):273–278, February 2003.

[119] Lecture notes from the MSc Course "Automated Planning" available online at : http://www.inf.ed.ac.uk/teaching/courses/plan/. Accessed Oct 2013.

[120] Dana Nau, Malik Ghallabl, and Paolo Traverso. *Automated Planning: Theory and Practice*. Elsevier, May 2004.

[121] Richard Fikes and Nils Nilsson. STRIPS: a new approach to the application of theorem proving to problem solving. *Artificial Intelligence*, 2(3-4):189–208, September 1971.

[122] Jörg Hoffmann and B. Nebel. The FF planning system : Fast plan generation through heuristic search. *Journal of Artificial Intelligence Research*, 14:253–302, 2001.

[123] Edwin Pednault. Adl and the state-transition model of action. *Journal of Logic and Computation*, 4(5):467–512, 1994.

[124] D. McDermott and The AIPS-98 Planning Competition Committee. PDDL-the planning domain definition language. Technical report, 1998. Available online at: http://www.cs.yale.edu/homes/dvm/. Accessed Oct 2013.

[125] Maria Fox and Derek Long. Modelling mixed discrete-continuous domains for planning. *Journal of Artificial Intelligence Research*, 27:235–297, October 2006.

[126] H. Younes and M. Littman. PPDDL1.0: An extension to PDDL for expressing planning domains with probabilistic effects. Technical report, Carnegie Mellon University, October 2004.

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

[128] Alfonso Gerevini and Derek Long. Preferences and soft constraints in PDDL 3. In *ICAPS-2006 Workshop on Preferences and Soft Constraints in Planning*, pages 46–54, 2006.

[129] The website for the 2nd IPC in 2000 can be found online at: http://www.cs.toronto.edu/aips2000/. Accessed Oct 2013.

[130] Derek Long and Maria Fox. The 3rd international planning competition: Results and analysis. *Journal of Artificial Intelligence Research*, 20:1 – 59, December 2003.

[131] Andrew Coles, Maria Fox, Keith Halsey, Derek Long, and Amanda Smith. Managing concurrency in temporal planning using planner-scheduler interaction. *Artificial Intelligence*, 173:1–44, 2009.

[132] William Cushing, Subbarao Kambhampati, and Daniel S. Weld. When is temporal planning really temporal? In *International Joint Conference on Artificial Intelligence (IJCAI)*, pages 1852–1859, 2007.

[133] Amanda Coles, Andrew Coles, Maria Fox, and Derek Long. COLIN : Planning with continuous linear numeric change. *Journal of Artificial Intelligence Research*, 44:1–96, May 2012.

[134] Andrew Coles, Maria Fox, Derek Long, and Amanda Smith. Planning with problems requiring temporal coordination. In *Proceedings of the 23rd national conference on Artificial intelligence - Volume 2*, AAAI'08, pages 892–897, 2008.

[135] The domains used in the Probabilistic track of IPC 4 can be found online: http://www.cs.rutgers.edu/ mlittman/topics/ipc04-pt/. Accessed Oct 2013.

[136] Alfonso Gerevini, Blai Bonet, and Bob Givan. Fifth international planning competition. Summary of participating planners in the 5th IPC in 2006. Available online at: http://www.ldc.usb.ve/ bonet/ipc5.

[137] T. Mitchell. *Machine Learning*. McGraw Hill, 1997.

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

[139] The website for the Learning Track of the 7th IPC in 2011, available online: http://www.plg.inf.uc3m.es/ipc2011-learning/. Accessed Oct 2013.

[140] Jesse Hoey and Marek Grzes. Distributed control of situated assistance in large domains with many tasks. In *Proceedings of the 21st International Conference on Automated Planning and Scheduling*. ICAPS, 2011.

[141] Ronald Petrick and Mary Foster. Planning for social interaction in a robot bartender domain. Awaiting Publication., 2013.

[142] Jörg Hoffmann and Stefan Edelkamp. The deterministic part of ipc-4: An overview. *Journal of Artificial Intelligence Research*, 24:519 – 579, 2005.

[143] H. Younes and R. Simmons. VHPOP: Versatile heuristic partial order planner. *Journal of Artificial Intelligence Research*, 20:405–430, 2003.

[144] Alfonso Gerevini and Ivan Serina. LPG: A planner based on local search for planning graphs with action costs. In *AIPS*, pages 13–22, 2002.

[145] Vincent Vidal and Héctor Geffner. Branching and pruning: An optimal temporal pocl planner based on constraint programming. *Artificial Intelligence*, 170:298 – 335, March 2006.

[146] Vincent Vidal. CPT4 : An optimal temporal planner lost in a planning competition without optimal temporal track. In *7th International Planning Competition (IPC-2011)*, pages 25–28, June 2011.

[147] G. Della Penna, B. Intrigila, D. Magazzeni, and F. Mercorio. UPMurphi: a tool for universal planning on PDDL+ problems. In *Proceedings of the 19th International Conference on Automated Planning and Scheduling*. ICAPS, 2009.

[148] Stefan Edelkamp. Taming numbers and durations in the model-checking integrated planning system. *Journal of Artificial Intelligence Research*, 20:195–238, 2003.

[149] Henry Kautz and Bart Selman. Planning as satisfiability. In *ECAI-92*, pages 359–362, 1992.

[150] J. Benton, Menkes van den Briel, and Subbarao Kambhampati. A hybrid linear programming and relaxed plan heuristic for partial satisfaction planning problems. In *Proceedings of the 17th International Conference on Automated Planning and Scheduling (ICAPS 2007)*, pages 34–41, 2007.

[151] Drew McDermott. The 1998 AI planning systems competition. *AI Magazine*, 21(2):35–55, 2000.

[152] F. Bacchus. The AIPS'00 planning competition. *AI Magazine*, 22(3):47–56, 2001.

[153] Alfonso E. Gerevini, Patrik Haslum, Derek Long, Alessandro Saetti, and Yannis Dimopoulos. Deterministic planning in the fifth international planning competition: PDDL3 and experimental evaluation of the planners. *Artificial Intelligence Journal*, 173:619 – 668, April 2009.

[154] ICAPS competitions. Index for all the International Planning Competition websites, available online at : http://ipc.icaps-conference.org/. Accessed Oct 2013.

[155] Jörg Hoffmann, Stefan Edelkamp, Sylvie Thiébaux, Roman Englert, Frederico Liporace, and Sebastian Trüeg. Engineering benchmarks for planning: the domains used in the deterministic part of IPC-4. *Journal of Artificial Intelligence Research*, 26:453 – 541, 2006.

[156] The full set of Domains for IPC 4 is available online at : http://www.tzi.de/ edelkamp/ipc-4/domains.tgz. Accessed Aug 2013.

[157] The full set of Domains for IPC 5 is available online at : http://zeus.ing.unibs.it/ipc-5/IPC5-domains.tgz. Accessed Aug 2013.

[158] The travelling and purchase problem domain, 2006. available at : http://zeus.ing.unibs.it/ipc-5/domain-descriptions/TPP.txt. Accessed Aug 2013.

[159] Y. Dimopoulos, A. Gerevini, and A. Saetti. The pathways domain, 2006. available at : http://zeus.ing.unibs.it/ipc-5/domain-descriptions/pathways.txt. Accessed Aug 2013.

[160] The trucks domain, 2006. available at : http://zeus.ing.unibs.it/ipc-5/domain-descriptions/trucks.txt. Accessed Aug 2013.

[161] Domain Descriptions for IPC 6, 2008. Available from : http://ipc.informatik.uni-freiburg.de/Domains. Accessed Aug 2013.

[162] The Nomystery Domain Description. Available at : http://www.plg.inf.uc3m.es/ipc2011-deterministic/DomainsTemporal#Nomystery. Accessed Aug 2013.

[163] The openstacks domain, 2006. available at : http://zeus.ing.unibs.it/ipc-5/domain-descriptions/openstacks.txt. Accessed Aug 2013.

[164] The domain and problem files for IPC 7 (2011) can be downloaded from the following svn repository : svn://svn@pleiades.plg.inf.uc3m.es/ipc2011/data/domains. Accessed Aug 2013.

[165] Results for IPC 6, 2008. Available from : http://zeus.ing.unibs.it/ipc-5/IPC5-results.tgz. Accessed Aug 2013.

[166] Brief description of the temporal domains used in the 7th International Planning Competition, available online at : http://www.plg.inf.uc3m.es/ipc2011-deterministic/DomainsTemporal. Accessed Aug 2013.

[167] Avrim L. Blum and Merrick L. Furst. Fast planning through planning graph analysis. *Artificial Intelligence*, 90(1):1636–1642, 1995.

[168] Patrik Haslum and Héctor Geffner. Admissible heuristics for optimal planning. In *Proceedings of the Fifth International Conference on AI Planning Systems*, pages 70 – 82. AAAI Press, 2000.

[169] Patrik Haslum. Improving heuristics through relaxed search - an analysis of TP4 and HSP*a in the 2004 planning competition. *Journal of Artificial Intelligence Research*, 25:233 – 267, February 2006.

[170] Henry Kautz and Bart Selman. Unifying sat-based and graph-based planning. In *International Joint Conference on Artificial Intelligence (IJCAI-99)*, volume 1, pages 318–325, 1999.

[171] Eric Parker. Making graphplan goal-directed. In *Proceedings of the Fifth European Conference on Planning (ECP-99)*, pages 333–346, September 1999.

[172] Antonio Garrido, Eva Onainda, and Federico Barber. A temporal planning system for time-optimal planning. In *In Proceedings of the Tenth Portuguese Conference on Artificial Intelligence*, pages 379–392. Springer, 2001.

[173] Patrik Haslum and Héctor Geffner. Heuristic planning with time and resources. In *Proceedings of the Sixth European Conference on Planning (ECP-01)*, September 2001.

[174] Drew Mcdermott. A heuristic estimator for means-ends analysis in planning. In *Third International Conference on AI Planning Systems (AIPS-96)*, pages 142–149, 1996.

[175] Blai Bonet, Gbor Loerincs, and Hctor Geffner. A robust and fast action selection mechanism for planning. In *14th National Conference of the American Association for Artificial Intelligence (AAAI-97)*, pages 714–719. MIT Press, 1997.

[176] Jörg Hoffmann. The Metric-FF planning system : Translating "ignoring delete lists" to numeric state variables. *Journal of Artificial Intelligence Research*, 20:291–341, December 2003.

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

[178] Craig Knoblock. Generating parallel execution plans with a partial-order planner. In *Proceedings of the 2nd International Conference on Artificial Intelligence Planning Systems (AIPS)*, pages 98–103, 1994.

[179] Malik Ghallab and Herve Laruelle. Representation and control in IxTeT, a temporal planner. In *Proceedings of the Second International Conference on Artificial Intelligence Planning Systems (AIPS-94)*, pages 61–67. AAAI Press, 1994.

[180] Derek Long and Maria Fox. Exploiting a graphplan framework in temporal planning. In *Proceedings of the 13th International Conference on Automated Planning and Scheduling (ICAPS)*, pages 52–61, 2003.

[181] Giuseppe Della Penna, Benedetto Intrigila, Daniele Magazzeni, and Fabio Mercorio. A PDDL+ benchmark problem: The batch chemical plant. In *Proceedings of the 21st International Conference on Automated Planning and Scheduling*. ICAPS, 2010.

[182] Hui X. Li and Brian C. Williams. Generative planning for hybrid systems based on flow tubes. In *Proceedings of the 18th International Conference on Automated Planning and Scheduling (ICAPS 2008)*, pages 206–213, 2008.

[183] The full set of Results for IPC 4 is available online at : http://www.tzi.de/ edelkamp/ipc-4/results-ascii.tgz. Accessed Aug 2013.

[184] Yixin 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, August 2006.

[185] The full set of Results for IPC 5 is available online at : http://zeus.ing.unibs.it/ipc-5/. Accessed Aug 2013.

[186] M Do, W Ruml, and R Zhou. On-line planning and scheduling: An application to controlling modular printers. In *Proceedings of the 23rd AAAI Conference on Artificial Intelligence*. Association for the Advancement of Artificial Intelligence (AAAI), 2008.

[187] Minh Do et al. Online planning for a material control system for liquid crystal display manufacturing. In *Proceedings of the 21st International Conference on Automated Planning and Scheduling*. ICAPS, 2011.

[188] Maria Fox, Derek Long, and Daniele Magazzeni. Automatic construction of efficient multiple battery usage policies. In *International Conference on Automated Planning and Scheduling (ICAPS)*, 2011.

[189] Daniel Kovacs. Complete BNF description of PDDL 3.1. Available online at : http://www.plg.inf.uc3m.es/ipc2011-deterministic/Resources. Accessed Aug 2013.

[190] A Description of how the results from IPC 7 are calculated and the results themselves can be found online at : http://www.plg.inf.uc3m.es/ipc2011-deterministic/Results. Accessed Aug 2013.

[191] 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, January 2009.

[192] Joshua Campion, Chris Dent, Maria Fox, Derek Long, and Daniele Magazzeni. Challenge: Modelling unit commitment as a planning problem. In *23rd International Conference on Automated Planning and Scheduling*. AAAI Publications, 2013. This paper has been accepted for publication but has not been published at time of writing. Is available online at: http://www.aaai.org/ocs/index.php/ICAPS/ICAPS13/paper/view/6041. Accessed Aug 2013.

[193] The National Grid, May 2012. [Online] : http://www.nationalgrid.com/uk/Electricity/Data/Demand+Data/.

[194] Data for the All Island Grid project detailing the Irish Generation portfolio. Available online: http://www.allislandproject.org/en/market_decision_documents.aspx?article=151a9561-cef9-47f2-9f48-21f6c62cef34. Accessed Oct 2012.