University of Strathclyde

Department of Computer and Information Sciences

Multi-objective Planning Using Linear Programming

By

Nor Haizan Mohamed Radzi

A thesis presented in fulfilment of the requirements for the degree
of Doctor of Philosophy
April 2010

This thesis is the result of the author's original research. It has been composed by the author and has not been previously submitted for examination which has led to the award of a degree.

Signed: Date :

To:

my husband; Shabudin Mat

my children; Nur Balqis, Nur Ain Afiqah, Muhammad Syahmi

and my parents

Acknowledgements

Upon the successful completion of this thesis, I would like to thank the greatest ALLAH for honoring me with the opportunity, patience, strength and perseverance to complete my study. I take this opportunity to express the deepest appreciation to my supervisors, Prof Maria Fox and Prof Derek Long, for their supervision, guidance, insight and continued support throughout the research. Without their guidance and persistence help this thesis would not have been possible. Many thanks go to the Malaysian Ministry of Higher Education and Universiti Teknologi Malaysia for funding my PhD.

I am indebted to many of my colleagues, particularly members of the Strathclyde Planning Group, present and past, who have shared their thought, knowledge and experience and kindly grant me their time even for answering some of my unintelligent questions. Thank you for your friendship, support and make my stay at the university a memorable and valuable experience.

And finally, my husband, children and parents; words cannot truly express how much I owe you. Thank you so much for your love, help and encouragement. This work is dedicated to all of you. I also offer my regards and blessing to all my family in Perak, Perlis and Kuala Lumpur who supported me in any respect during the completion of my research and keep praying for my success in this journey.

Contents

1 Introduction **1**

1.1 Domain-Independent and Classical Planning 1
1.2 Planning versus Scheduling 3
1.3 Metric Planning 4
1.4 Scope, Motivation and Objective 6
1.5 Thesis Outline 9

2 Background **12**

2.1 Metric Planning 12
2.2 PDDL2.1 13
2.2.1 Plan Metric 13
2.3 IPC3 Numeric Domains 17
2.4 Numeric Domains Analysis 19
2.5 Approaches in Planning Systems 22
2.6 Numeric Planners 23
2.7 Planning and Scheduling Problem 25
2.7.1 Integrating Planning and Scheduling 28
2.8 Multi-objective Planning 29
2.9 Multi-objective Functions in Linear Programming 30
2.10 Conclusion 33

3 Domain-Specific Metric **34**

3.1 Introduction 34
3.2 Characteristics of the Planning Domains in IPCs Series 35
3.3 New Domain Class 36
3.3.1 Extended Settler Domain 38
3.3.2 Trader Domain 40
3.3.3 Bread Domain 41
3.3.4 Production Domain 44

3.3.5 Sugar-Supply Domain 46
3.4 Conclusion 48

4 Background To LPRPG **51**
4.1 Introduction 51
4.2 Metric-FF 52
4.3 LRPPG 56
4.3.1 Relaxed Planning Graph Construction 57
4.3.2 Relaxed Planning Graph Extraction 59
4.4 Conclusion 61

5 MetricLPRPG **62**
5.1 Introduction 62
5.2 Construction of Multiple Solutions in Metric Domain 63
5.3 Implementation of the Plan Metric in MetricLPRPG Heuristic 66
5.3.1 Including the Plan Metric Variables in Relaxed Plan Expansion . 67
5.3.2 Multi-objective Linear Programming in Relaxed Plan Extraction 70
5.4 Conclusion 76

6 Results **79**
6.1 Introduction 79
6.2 Planners and Experiment Considerations 79
6.2.1 Extended Settler Domain 81
6.2.2 Bread Domain 85
6.2.3 Production Domain 90
6.2.4 Trader Domain 94
6.2.5 Sugar Domain 97
6.3 Conclusion 99

7 Conclusion **105**
7.1 Summary 105
7.2 Limitation 107
7.3 Future Work 109

A **112**

B **121**

C **126**

D **131**

E **141**

List of Figures

1.1 Search Space in Planning Problem 4
1.2 Search Space in Scheduling Problem 5
1.3 Search Space in Metric Planning Problem 7

2.1 Turn-to Action in Satellite Domain 14
2.2 Action Sequences in Satellite Domain 20
2.3 Action Sequences in Rover Domain 22
2.4 Resource and Action Choices Continuum in Planning and Scheduling . . 27
2.5 Metric Planning Action Choices Flows 28

3.1 Example of metric function in Domain-specific metric problem 37
3.2 Alternative Actions that Achieve Subgoal *housing location0* ≥ 0 in Extended Settler Domain 40
3.3 General Action Sequences in Trader Domain 41
3.4 General Action Sequences in Bread Domain 42
3.5 Solution with Plan Metric *labour* and *energy* in Bread Domain 43
3.6 Solution with Plan Metric *energy* and *pollution* in Bread Domain 44
3.7 General Action Sequences in Production Domain 46
3.8 General Action Sequences in Sugar Domain 49

4.1 Simple Supply Chain Domain 54
4.2 Bounds Propagation in the Relaxed Planning Graph for Supply Chain Domain 54
4.3 LP Estimation for the First Layer in the Relaxed Planning Graph . . . 58
4.4 LP Model for numeric variable ft_1 at layer 1 58
4.5 LP Model for Numeric Variable tk_1 at Layer 1 58
4.6 LP Estimation for all Numeric Variables in all Layers of the Relaxed Planning Graph in Supply Chain Domain 59
4.7 LP Model for Numeric Variable ft_3 at layer 3 in Supply Chain Domain 59
4.8 Numeric Constraints at Layer 2 and 3 in Supply Chain Domain 60

4.9 LP Model for Relaxed Plan Extraction in Supply Chain Domain 61

5.1 Alternative Actions in Extended Settler Domain 64
5.2 Existing Actions in Settlers Domain 65
5.3 Potential Relaxed Plan for Extended Settler(1) 65
5.4 Potential Relaxed Plan for Extended Settler(2) 65
5.5 Potential Relaxed Plan for Extended Settler(3) 65
5.6 Relaxed Planning Graph for Extended Settler at Layer 1 68
5.7 General constraint for numeric variables in Extended Settler Domain . 68
5.8 LP Model for Variable *pollution* at Layer 1 for the Extended Settler Domain 68
5.9 LP Model for Variable *resource_use* at layer 1 for the Extended Settler Domain 69
5.10 LP Model for Variable *labour* at layer 1 for the Extended Settler Domain 69
5.11 Complete Relaxed Planning Graph for Extended Settler Domain 71
5.12 LP Model for Plan Extraction in the Extended Settler Domain 73
5.13 Relaxed Plan Constructed for Example 1 74
5.14 Relaxed Plan Constructed For example 2 75
5.15 Relaxed Plan Constructed in LPRPG heuristic 76

6.1 Plan Cost in Extended Settler Domain 83
6.2 Plan Length in Extended Settler Domain 83
6.3 Plan Cost in Bread Domain 89
6.4 Plan Length in Bread Domain 89
6.5 Plan Cost in Production domain 93
6.6 Plan Length in Production domain 93
6.7 Plan Cost in Trader Domain 97
6.8 Plan Length in Trader Domain 98
6.9 Plan Cost in Sugar Domain 102

List of Tables

3.1 Extended Settler plan metric profiles 39
3.2 Trader Domain plan metric profiles 42
3.3 Numeric preconditions and effects in the Bread Domain 43
3.4 Bread Domain plan metric profiles 43
3.5 Production Domain plan metric profiles 47
3.6 Numeric preconditions and effects in Production Domain 48
3.7 Numeric Preconditions and Effects in Sugar Domain 50

4.1 Difference Between Bounds Propagation and Manual Calculation 55

5.1 Notations for Variables 69
5.2 Manual Value Assignment of Action Variables 78

6.1 Results in the Extended Settler Domain 82
6.2 Percentage of Plan cost Reduction in MetricLPRPG in Extended Settlers domain 84
6.3 Percentage of Plan length Increasing in MetricLPRPG in Extended Settler Domain 86
6.4 Results in the Bread Domain - Plan Cost 87
6.5 Results in the Bread Domain - Plan length 88
6.6 Percentage of cost reduction in MetricLPRPG vs other planner in Bread domain 90
6.7 Percentage of length increment in MetricLPRPG vs other planners in Bread Domain 91
6.8 Results in the Production Domain 92
6.9 Percentage of cost reduction given by MetricLPRPG in the Production Domain 94
6.10 Percentage of length increasing given by MetricLPRPG in the Production Domain 95
6.11 Results in the Trader Domain 96

6.12 Percentage of cost reduction in MetricLPRPG in Trader Domain 99
6.13 Percentage of plan length increasing in MetricLPRPG compared to other planners in the Trader Domain 100
6.14 Results in the Sugar Domain 101
6.15 Percentage of Cost reduction in MetricLPRPG against other planners in Sugar Domain 103
6.16 Percentage of length increment in MetricLPRPG against other planners in Sugar Domain 104

Abstract

Metric planning allows planning problems with numerical resources to be efficiently modelled and solved. Such problems arise in many real world planning applications. The plan metric, the optimisation function introduced in metric planning, is designed to facilitate resource optimisation usage in the developed solutions. However, most published benchmark domains do not fully exploit the expressiveness of the plan metric functions and are therefore limited in terms of the resource optimisation problems that they pose. In particular, the domains do not provide choices of resources that can influence the value of plan metric. Therefore, many numeric planners ignore the optimisation of the plan metric and simplify the solution development by focusing on minimising the plan length.

This thesis contributes to a comprehensive analysis of the structure of a new class of metric domain called *domain-specific metric planning*. The domain-specific metric problems exploit the ability of the plan metric to include aspects of planning and resource scheduling. The domain is enriched with resource choices that change the value of the plan metric and result in the generation of different possible plans. The generated plans are different in terms of the value of both plan length and plan metric. The trade-off between the plan length and plan metric values is demonstrated in the solution of domain-specific metric problems.

MetricLPRPG, the planner written in this thesis, extends the previously published LPRPG planner by including the plan metric in its heuristic. The novel heuristic implemented in MetricLPRPG uses Multi-Objective Linear Programming to minimise both values of plan length and plan metric simultaneously. This heuristic provides a basis for choosing actions with the minimum resource cost with regard to the encoded plan metric to be incorporated in the relaxed plan.

Results obtained show MetricLPRPG to be a competitive planner that achieves a *compromise* between minimising the plan length and minimising the value of the plan metric. The quality of the solutions obtained is interestingly different from the quality of solutions obtained by LPRPG. MetricLPRPG can produce better quality plans (though sometimes at the cost of increased length) particularly in solutions developed for problems in the domain-specific metric class.

Chapter 1

Introduction

1.1 Domain-Independent and Classical Planning

Domain-independent planning technology is motivated towards solving real world planning problems. The problem input consists of a description of the world initial state, a set of operators that perform state transitions and a goal state. Since, it is usually impossible to include an encoding of all possible states and states transitions, domain-independent planning relies on an abstract, general model of actions [35]. The simplest form of the domain-independent planning problem is known as *classical planning*. The domain is encoded in propositional logic, typically using STRIPS [21]. In STRIPS, the states of the world are described together with the set of operators which transform the current world states to other states. The new states are generated from previous states by removing all the propositions present in the delete effects list and adding the propositions in the add effects list only if all the precondition propositions are true in that state. However STRIPS, only considers conjunctive predicates. An extended version of STRIPS, called ADL [51], increases the convenience of domain encoding by extending the syntax to allow disjunctive, quantified and negative preconditions. The syntax for describing the effects of actions is also extended to allow conditional effects; effects which are applied whenever a given condition holds.

Despite these extensions, some assumptions hold in classical planning in order to simplify solution approaches by planners. The assumptions include the following [61]:

- atomic time: a single unit time is required to execute an action and this time is indivisible
- deterministic: all states in a described world are fully-observable, static and deterministic. Transformation from one state to another state of the world is only

performed by the application of actions. The effects of actions are also deterministic.

- no representation of resources: a resource requirement or consumption cannot be specified, for example to describe the amount of fuel required for turning a spacecraft or the amount of data storage or power required for taking an image.[56]

These assumptions have weakened the capability to model various essential phenomena in real world planning problems. In particular, the representation of resources, which is explored in this thesis. Almost every practical application is influenced by the use of resources. Examples of resources include money, energy, labour and machines. Different actions in the problem might require different consumption of resources. It is possible to encode the levels or quantities of resources in classical planning, but it would be necessary to discretise all quantities and expand a completely grounded representation. This approach is rather tedious and limits some aspects of modelling quantities. For example, in cases where resource consumption is a function of other values, this cannot be modelled. Furthermore, including resource constraints in the problem requires a means for measuring resource efficiency. For instance, the plan that uses the minimum cost of resources.

In order to meet these requirements, significant extensions have been made to the Planning Domain Description Language, PDDL2.1 [24] by introducing new encoding schemes called metric fluents and a language for expressing a plan metric. Metric fluents are variables that can take numeric values and can be used to represent resource modelling more conveniently, including resource quantities and constraints. The plan metric acts as an objective function enabling optimisation over selected numeric variables, usually those representing the critical resources. The extension to support numbers in classical planning is called Numeric or Metric Planning. These extensions allow the easy modelling of resources but increase the complexity of solution development particularly if a plan metric is used. For example in modelling resources in a supply chain planning problem in manufacturing. Optimising the resource usage is important in planning the activities involved in the supply chain. In this kind of problem, minimising the number of actions, which is the mandatory requirement for good solution in classical planning, is not the only objective. In fact, the optimisation of resources makes it possible to increase the number of actions in a solution. This is due to the fact that a low cost plan might require multiple steps using low cost resources. In other words, there will be a trade-off between the plan length and plan cost in the solution. Many state-of-art numeric planners deal with numeric preconditions as they are usually used to represent the resource constraints, and do not take into account

the plan metric in the solutions. Furthermore, the plan metric is not required to find a feasible solution. The trade off between minimising the number of actions and optimising over particular variables in metric planning is a central focus in this thesis. The extended metric planner developed in this thesis attempts to produce reasonable plans with respect to both plan length and plan metric.

1.2 Planning versus Scheduling

Planning involves selecting and organising actions to achieve a desired goal and satisfy a set of domain constraints [25]. Most planning problems are combinatorial in the sense that the number of action choices, or the time required to evaluate a given course of action, is exponential in the description of the problem [11]. The tree-like structure in Figure 1.1 illustrates the massive search space of action choices and their causal inter-dependencies. As shown in the Figure, each selected action can lead to different action sequences. If the heuristic value is not appropriately calculated or uninformative, it will lead to a dead-end or to no solution. Planning technology has traditionally concentrated on finding feasible solutions, measuring quality by the number of actions in the generated plan. In other words, planning is concerned with finding *what* actions are required in a feasible solution.

In contrast, in traditional Scheduling action selection is eliminated, as the action sequence is already defined or committed to in most scheduling problems. The main task is to allocate the limited resources to actions and retain a valid plan [30]. Some cases of scheduling have action choices, particularly in the manufacturing domain where resources and times are assigned to actions so that they conform to the constraints on the schedule [25]. But, it is easier to allocate resources to an existing activity sequence than it is to choose between multiple different actions. Furthermore, the heart of the scheduling problem is allocation of resources. The resource allocation task in scheduling is typically required to be optimal. Hence, the scheduling problem is viewed as an optimisation problem, but, as explained feasibility is more important than optimisation in planning [56]. Furthermore both planning and scheduling are different in terms of complexity of finding the solution. Job Shop Scheduling is categorised as NP equivalent [26]. However, planning is harder to solve compared to job shop scheduling problem and it is categorised as PSPACE-complete even if some severe restrictions are applied [7].

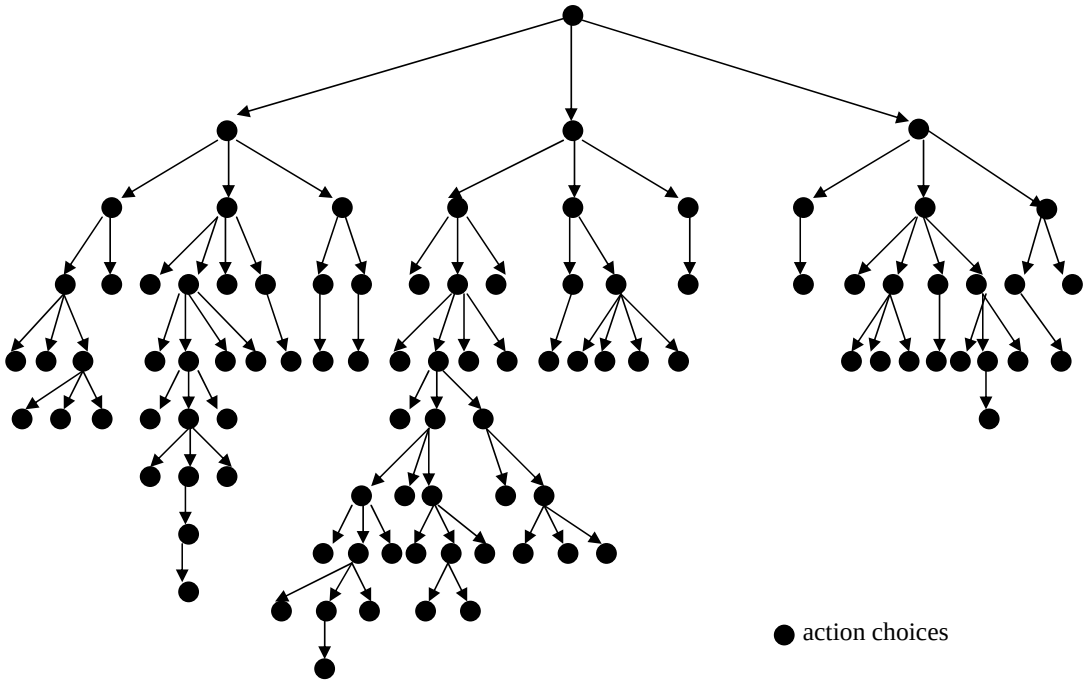

Figure 1.1: Search Space in Planning Problem

The tree-like structure in Figure 1.2 shows the action choices and the resource allocation activities in scheduling. There are far fewer action nodes than arise in a planning problem. If the tree-like structure in Figure 1.1 can be called a *forest*, the tree in Figure 1.2 is a *tree*, or *sub-forest*. As shown in the figure, different allocations of resources and time slots to the available action choices does not have a big impact on the action sequences. Therefore for scheduling, the criteria by which solutions are judged can vary: minimising the use of resources, minimising the total time for the whole schedule or maximising the slack periods are common strategies. The focus of this problem in developing a solution is to find *when* and *what* resources the action should use in the schedule.

1.3 Metric Planning

Metric planning is an extended version of classical planning where new encoding schemes are added into the PDDL2.1 language to make the modeling of quantities more efficient. Hence in the metric planning problem, facts have not only true or false conditions but also quantitative conditions. This extension makes the planning technology more applicable in modelling real applications particularly for resource-intensive problems. In

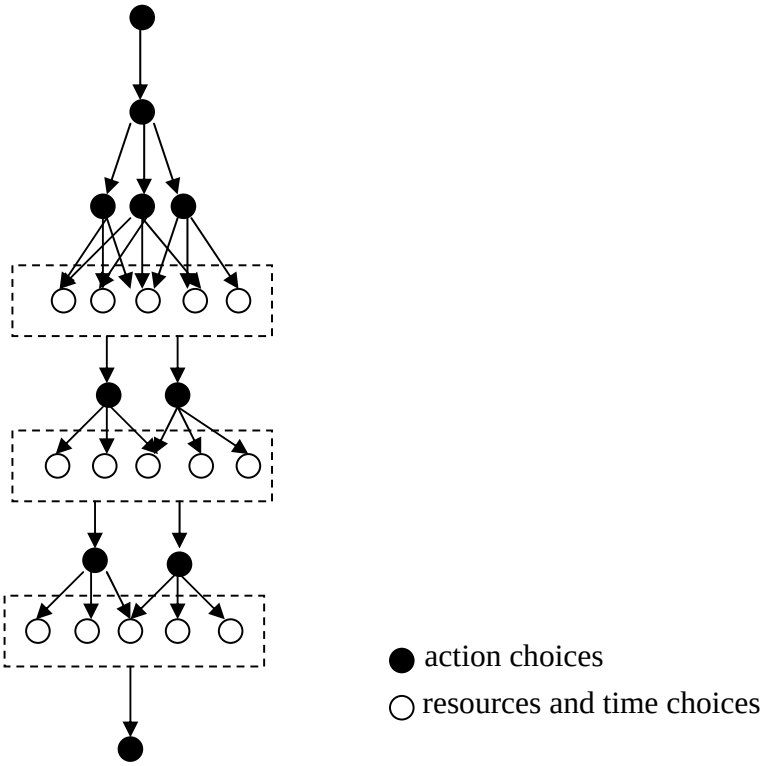

Figure 1.2: Search Space in Scheduling Problem

these problems, action execution is generally constrained by the resource quantities.

In the metric planning domain, a numeric fluent or variable can be encoded in the action preconditions and effects. The numeric precondition is used to represent a resource constraint in a particular action. For example, the numeric precondition *(fuel* $\geq$ *(distance ?c1 ?c2)*(slow-burn ?a))* is encoded in the *fly* action in the ZenoTravel domain in IPC3 [46]. The action is executed only if the condition is true. The value of a numeric fluent is then updated in the action effect or after the action execution to indicate the resource consumption. For example in the *fly* action, the updated value of the numeric resources is accomplished by statement; *(decrease (fuel ?a) (* *(distance ?c1 ?c2) (slow-burn ?a)))))*, as encoded in the action effects. Furthermore, the numeric constraints are also allowed in the goals, in which the value of a numeric fluent or variable is either at least as high as or at most as high as a given constant [24].

Another significant extension in PDDL 2.1 is the provision of the plan metrics [24]. The plan metric can be said to be an optimisation or objective function. The objective function provides a new means of plan evaluation which is the plan quality. With the plan metric and a problem with similar initial and goal states, it would be possible to produce a few solutions which are completely different in terms of the plan value. Therefore, the plan metric allows the domain modeller to experiment with the effect of different sequences of actions with the different metric. The plan metric extension enhances the expressiveness of the language to model the resource optimisation that is necessary in almost all resource intensive planning domains. Including resource optimisation in planning also brings the scheduling element into the classical planning problem. The problems in planning and scheduling can lie in a continuum which is one end is pure planning and another end is pure scheduling. At the pure planning end, the concern is with logical reasoning, since the problems are rich with action choices. Whilst at pure scheduling end, the problems are rich with resource choices and the concern is with exact allocation of the resources. Metric planning with resource optimisation, however, lies in the middle of the continuum. Therefore, the problem is now enriched with choices of action as well as choices of resources. Figure 1.3 illustrates the search space and resource allocation in metric planning with resource optimisation. The search space is far more complex compared to the search space in Figure 1.1 and Figure 1.2. The development of the solution becomes harder as it must not only seek for a feasible but also an optimal solution.

Although incorporating the plan metric in the problem is optional, it is necessary for many real world applications. But, taking the plan metric into the solution development can result in many problems with feasible solutions failing to be solved. Therefore, many state-of-art numeric planners such as Metric-FF [38], SGPlan [8], LPG [33] ignore the plan metric and develop solutions that satisfy only the logical goals. Having realised that many practical applications require a solution reflecting the plan metric, the planner developed in this thesis uses the plan metric information in the problem to develop a heuristic which is later used to guide search towards a solution. The planner has employed a very well known optimisation tool, linear programming, to obtain the heuristic during the construction of the relaxed planning graph.

1.4 Scope, Motivation and Objective

The scope of this thesis is to examine a well-defined class of problems characterised by metric resources and a plan metric which is treated as an objective function to guide search toward high quality solutions. This class is defined as a *domain-specific*

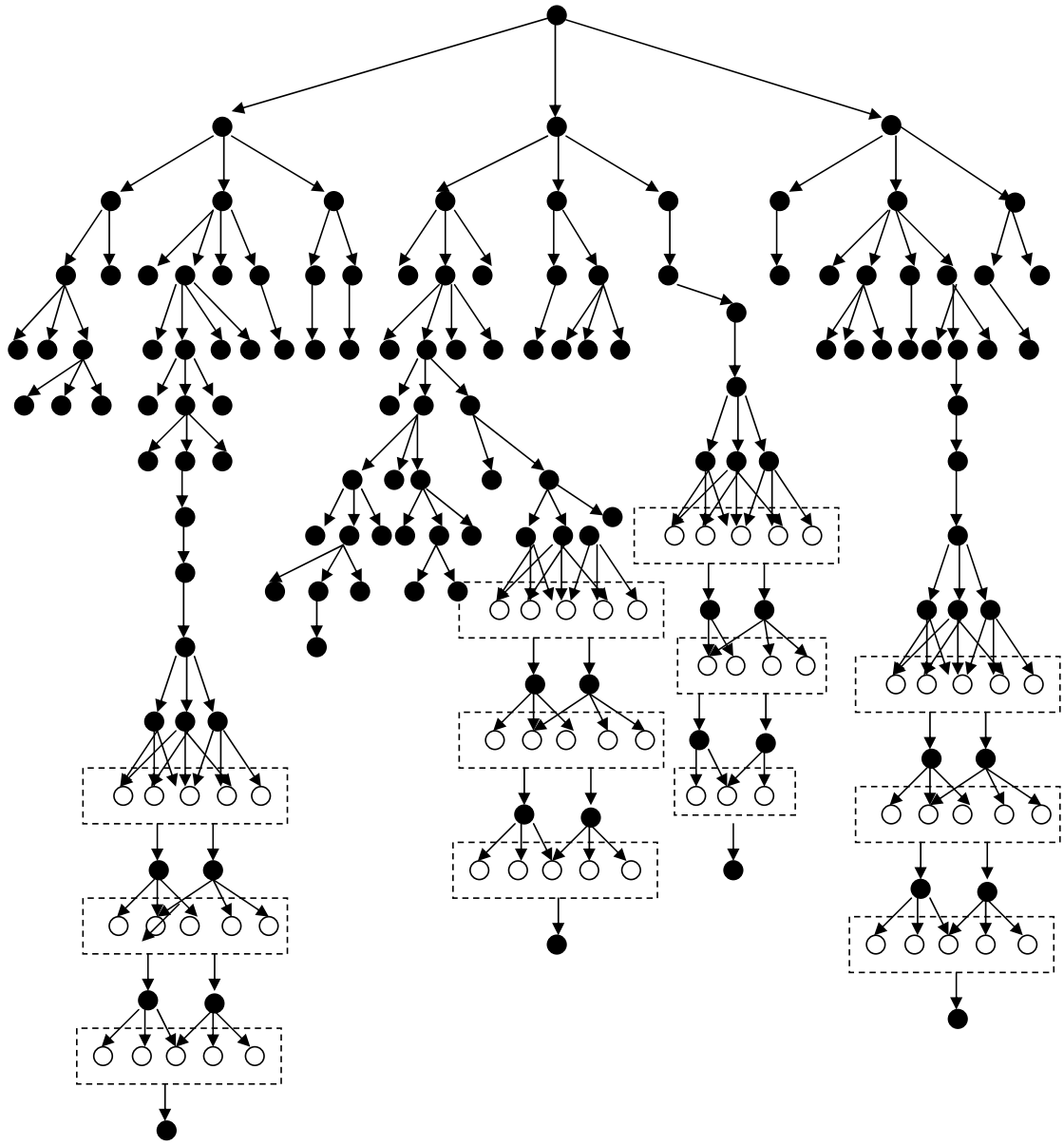

Figure 1.3: Search Space in Metric Planning Problem

metric class. The domain-specific metric class structure is blended with planning, scheduling and constraints reasoning. As a result, the problem structure is rich with numerical facts and constraints compared to the standard planning domain. At the

same time, it is also rich with logical facts and constraints which are nearly absent in the scheduling problem. The example of real world application of the domain-specific metric is planning for the manufacturing supply chain. The general structure of the problem comprises of planning and scheduling elements. Therefore, the solution developed for such a problem combines the decision about *what action* and *what resource* choices to make, resulting in different values of the objective function. This domain is also rich with numeric interactions whereby the numeric variables do not simply model the increasing and decreasing effects of resources but force constraints on other numeric variables (eg. a numeric resource used to develop another numeric resource).

The motivation of this work is that the domain-specific metric class has so far been largely ignored by the planning community. This kind of problem exists in many real world applications and attempts to solve them will increase the applicability of planning technology to real world problems. Although the plan metric is defined in some previous benchmark domains [12, 46] they are not properly used to guide choices between actions that lead to alternative solutions with different metric values. Numeric interaction is rarely considered, except in the Settlers [46] domain and in some domains not used in the competition (eg: Market). Most of the benchmark metric problems use numeric fluents to model simple infinite resources like fuel or energy consumption in action effects. Investigation of these domains is discussed in detail in Chapter 2.

The interaction between *what action* and *what resources* contributes to a trade-off between plan length and plan cost values in the developed solution. While currently there are a number of metric planners that have been developed, none of them are able to deal with this trade-off. Most of the planners only handle numeric preconditions, because numeric preconditions are critical in action selection. For example Metric-FF, a top performing numeric planner in IPC3 ignores the plan metric in its solution development. The plan metric is included only if it has been encoded as the numeric precondition of actions. Since the aim of the implemented heuristic is to minimise the number of actions, therefore, the minimum value of the plan metric turns out to coincide with minimising the number of actions in the plan. A series of experiments was conducted to investigate the performance of Metric-FF and some other numeric planners on metric problems with specific objective functions. The results obtained discovered that numeric planners including Metric-FF, LPG-td and some others in the literature either fail to produce any plan or only produce results for small problems and ignore the plan cost in the solution. These results are shown in Chapter 6.

The objective of this thesis is to develop a competitive planner to solve this problem by incorporating both aspects of plan length and plan cost in constructing the solu-

tion. It is achieved through extensions made to a planner called LPRPG [10]. LPRPG is a numeric planner developed with the aim of solving problems caused by the complex numeric interactions. It uses Linear Programming (LP) to determine accurate bounds on numeric precondition values during the construction of the relaxed planning graph. Metric-LPRPG, the extended planner, applies the same principles as LPRPG but extends the numeric variables in the LP model so that it includes the plan metric variables. This means the bounds and constraints for the plan metric variables are estimated and constructed along-side the numeric preconditions across the construction of the relaxed planning graph in Metric-LPRPG. Whilst LPRPG cannot make use of a domain-specific objective function, Metric-LPRPG is designed to do so. Managing the domain specific objective function necessitates extensions to all aspects of the relaxed graph construction and the extraction of the relaxed plan. As a consequence of these extensions Metric-LPRPG can arrive at solutions that trade plan length off against plan cost, producing plans exhibiting a reasonable balance between the two, whilst LPRPG could only attempt to minimise plan length.

A Linear program is constructed at solution extraction in both LPRPG and Metric-LPRPG. In LPRPG, the variables in the objective function only include the action variables. So that, the LP solution will give the minimum number of actions that achieve all numeric preconditions modelled in the LP. This implies LPRPG emphasises having the shortest plan, similar to other numeric planners but with more informed resources. Whilst in Metric-LPRPG, the plan metric variables are added to the objective function. Therefore, the LP solution is to minimise both action and plan metric variables.

The novel contributions of this work to the community include:

- An understanding of the structure of domains with specific objective functions
- A planner, Metric-LPRPG, that takes into account the plan metric which has been mainly ignored by the community, resulting in better quality plans.
- A novel heuristic that combines both plan length and plan cost

1.5 Thesis Outline

The next chapter covers the discussion of the research background. It defines the metric planning problems and explains those extended language features in PDDL2.1 that provide a more convenient way to model numeric resources present in most practical planning problems. The explanation includes the plan metric, the core issues tackled

in the thesis, which allows a plan to be measured in terms of quality. Four new characterisations of the expressiveness of planning domains with domain-specific plan metrics are identified and explained in Section 2.2.1.

Section 2.5 and 2.6 give a brief discussion of some numeric planners and approaches taken in dealing with numeric problems. This section also highlights that the plan metric is not included in developing a solution in the numeric planner such as Metric-FF, LPG and MIPs. In Section 2.7 and the differences between planning and scheduling problems are explained and highlighted. Most practical problems lie in the middle of the continuum between planning and scheduling. The features of the middle-continuum problem are found to be similar to most of the domain-specific metric class problems. This section also includes an abstract discussion of the research approaches that have been taken to solve problems with planning and scheduling elements. The two last sections of this chapter cover the discussion on Multi-objective Planning and Multi-objective Linear Programming. The Multi-objective Linear Programming is the optimisation tool adopted in the heuristic developed in this thesis.

Chapter 3 develops a new class of planning domains and problems that are considered a domain-specific metric class of problems and solved with the technology developed in the thesis. The new domains designed for the thesis include *Extended Settlers* which is the extended version of Settlers in IPC3, *Trader*, *Bread*, *Production* and *Sugar* domains. The discussion describes how the resource choices designed in the new domains affect the plan metrics. It also explains how different action sequences with different costs may be constructed as alternative solutions to problems within the domains.

Chapter 4 gives background on the LPRPG planner, the planner extended in this thesis. This chapter discusses the approach taken by the Metric-FF planner in estimating values of numeric variables during the construction of the relaxed planning graph. It also explains the implications of this approach particularly for domains that intensively use numeric variables. It explains how LPRPG overcomes problems that arise in Metric-FF by exploiting Linear Programming to obtain better heuristic values and better estimates of the values of numeric variables.

Chapter 5 describes the implementation of Metric-LPRPG, the extended planner developed in the thesis. It explains how the plan metric is included in the relaxed planning graph construction. The explanation also includes discussion on the implementation of Multi-objective Linear Programming in the relaxed plan extraction in order to obtain more informative heuristic values. The last section in this chapter

demonstrates the different between action selection with and without incorporating the plan metric.

Chapter 6 discusses and analyses results obtained from the implemented system by comparison with the results of LPRPG and some other selected state-of-the-art numeric planners using the new problems discussed in Chapter 3.

Finally, Chapter 7 summarises the work presented in the thesis and comments on the strengths and limitations of the approach taken. It also includes discussion of possible future work that could be considered to improve the implemented system.

Chapter 2

Background

2.1 Metric Planning

Metric planning is defined as a planning problem where a set of numeric variables, $V = \{v^1, ..., v^n\}$, is added to the problem. Notation for the numeric planning task can be written as a tuple $\langle V, P, A, I, G \rangle$ where P, A, I and G are defined similar to the STRIPS planning task [38]. Thus, states in the numeric planning task can have numeric and propositional state variables. This development is important and significant in order to make planning technologies more applicable to model realistic planning problems. Almost all of the real world planning problems use resources to execute the actions. The resources can include money, fuel or energy. Numbers, which can be represented by the numeric variables, provide a natural platform to model resources including their level or quantities and also their constraints.

The metric planning problem encoding is accomplished by the extensions made in the Planning Domain Description Language version 2.1 (PDDL2.1)[24]. This language was first used and tested in the Third International Planning Competition (IPC3) [46]. The syntax for expressing numeric constraints in PDDL2.1 is based on the previous version of PDDL, the language produced for the AIPS-98 Planning Competition [34]. However, the numeric syntax of PDDL was not tested either in AIPS-98 [49] or in the following International Planning Competition, IPC'00 [2]. Numeric domains and technologies were first tested and benchmarked in the third International Planning Competition in 2002 (known as IPC3).

2.2 PDDL2.1

The numeric extension allows the expression of numeric conditions and effects. In PDDL2.1, a numeric expression is constructed, using arithmetic operators, from primitive numeric expressions [24]. Numeric expressions can be employed as both effects and conditions of actions. In effects, operators like increase, decrease, scale up, scale down and assign can be used to update the value of a fluent by some function $(+, -, \times, \div)$ of fluents and real numbers. Conditions on the numeric expression are always comparisons between pairs of numeric expressions. Conditional operators $(\leq, \geq, <, >, =)$ can be used between numeric expressions and employed in the condition. These new features make the language more expressive enabling the modelling of quantities and constraints on the resources. With these features, the resource is not only described as present or absent, as in the propositional domain, but it also can be measured with quantitative terms, such as level of resource. Therefore, the discretisation of all possible quantities of the resources is no longer required as changes in resource availability can be efficiently handled by the new language features.

2.2.1 Plan Metric

The plan metric is a new field provided by PDDL2.1 [24]. It is an optional field and can be specified within the problem file. The plan metric plays a role like an objective function. It allows the domain designers to encode alternative solutions for a similar domain each giving a different value to the objective function. Good solutions are those that optimise the value of the plan metric. Thus, the plan metric gives rise to a new dimension of the plan evaluation called *metric plan quality*. In order to be able to exploit the metric during search, it has to be *instrumented* which means it must first initialise a value and then this value is updated in the domain description as planning progress[24]. For example, the plan metric defined in the *Satellite* domain in IPC3 [46] is as follows,

(:metric minimise (fuel_used))

The plan metric is initialised by *(=(fuel_used)0)* statement in the problem file. The value is then updated as a positive effect in the action called *turn_to*, which consumes fuel. This is shown in Figure 2.1

Without the above instrumentation, the minimum value of the plan metric cannot be achieved. Furthermore, the plan metric must be carefully instrumented in the domain. For example, a new action called *refuel* is added to the *Satellite*. It updates the fuel value in a similar way to the *turn_to* action but with different quantities. The plan

```
 (:action turn_to
   :parameters (?s - satellite ?d_new - direction ?d_prev - direction)
   :precondition (and (pointing ?s ?d_prev)
                      (not (= ?d_new ?d_prev))
(>= (fuel ?s) (slew_time ?d_new ?d_prev))
                )
   :effect (and  (pointing ?s ?d_new)
                 (not (pointing ?s ?d_prev))
(decrease (fuel ?s) (slew_time ?d_new ?d_prev))
(increase (fuel-used) (slew_time ?d_new ?d_prev))
           )
  )
```

Figure 2.1: Turn-to Action in Satellite Domain

metric is now changed to maximise the *fuel_used*. According to this, the maximum value of the plan metric can be achieved by including more *refuel* actions although these actions do not actually contribute to achieving the goals. But, the plan metric value can be very high if the constraint such as maximum capacity of the fuel tank or the available capacity of fuel is not encoded in the domain. In this case, the resource is considered infinite. Therefore, the optimal solution does not exist and the problem is said to be an ill-defined plan metric problem.

Besides enhancing the flexibility in domain modelling, the plan metric can result in a dramatic impact on the solution development. This is due to the interacting factors that often exist in many practical applications [24]. The interacting factors can include the fact that some of the actions might improve the plan quality whilst some other actions might worsen the plan quality [24]. The style of actions and plan metrics encoding in the domain and their implications for the solution development, particularly to obtain the optimal value of the plan metric, is examined in the following discussion. The term *metric function* is used to refer to the plan metric. Planning problems with metric function encodings can be classified according to these definitions: *strictly straightforward*, *straightforward*, *semi-straightforward* and *expressive*. These definitions are as follows:-

Definition 1 (Strictly straightforward metric function) *Given a planning problem* $\Phi = \langle V, P, A, I, G \rangle$*, a metric objective function* $f : Plan_\Phi \to \mathbb{R}$ *is* strictly straightforward *if, for any pair of plans for* Φ*,* π_1 *and* π_2*, such that* $f(\pi_1) < f(\pi_2)$*,* $|\pi_1| < |\pi_2|$*.*

If a metric function is *strictly straightforward* then minimising the length of the plan guarantees an optimal value of the function.

Definition 2 (Straightforward metric function) *Given a planning problem* $\Phi = \langle V, P, A, I, G \rangle$*, a metric objective function* $f : Plan_\Phi \to \mathbb{R}$ *is* straightforward *if, for any pair of plans for* Φ*,* π_1 *and* π_2*, such that* $f(\pi_1) \leq f(\pi_2)$*,* $|\pi_1| \leq |\pi_2|$*.*

If a metric function is *straightforward* then minimising the length of the plan does not guarantee an optimal value of the function, but no longer plan can possibly have a better metric value. This means that there could be multiple plans of the same length with different metric values. Thus, when searching for a plan in a fixed size structure (such as a plan graph), once a plan has been found it is guaranteed that there exists an optimal plan in the same sized structure.

Definition 3 (Semi-straightforward metric function) *Given a planning problem* $\Phi = \langle V, P, A, I, G \rangle$*, a metric objective function* $f : Plan_\Phi \to \mathbb{R}$ *is* semi-straightforward *if, for any pair of executable sequences of actions in* Φ*,* π_1 *and* π_2*, such that* π_1 *is a prefix of* π_2*,* $f(\pi_1) \leq f(\pi_2)$*.*

If a metric function is *semi-straightforward* then minimising the length of the plan does not guarantee an optimal value of the function. However, because actions never decrease the cost it will always be a good strategy to minimise the number of actions. This would not be the case if actions could *decrease* the metric value of the plan during its construction.

Definition 4 (Expressive metric function) *Given a planning problem* $\Phi = \langle V, P, A, I, G \rangle$*, a metric objective function* $f : Plan_\Phi \to \mathbb{R}$ *is* expressive *if* f *is not strictly straightforward or straightforward.*

If a metric function is *expressive* then minimising the length of the plan does not guarantee an optimal value of the function. Adding more actions to a plan might decrease its metric cost.

How can a planning problem with a metric function be defined as *strictly straightforward*?. The metric function in the above *Satellite* problem is about minimising the fuel used. The *turn_to* action increases the *fuel-used* value. This can be seen from Figure 2.1

where the fuel is encoded as the numeric precondition and positive effect in that action. This means, the metric function value increases, as the number of *turn_to* actions increases in the plan. As a consequence, the quality of plan decreases. Good quality solutions are supposed to include the minimum number of *turn_to* actions. As a result of minimising the number of these actions, the metric function value will be minimum. In this case, the metric function in this problem is defined as *strictly straightforward* since the optimal value of the metric function is obtained as a result of minimising the number of actions in the plan. Most of the previous benchmarked numeric domains, particularly those in IPC3, fall within this category. The detailed discussion of these domains is covered in Section 2.4.

Suppose, an action called *turn_to_battery_power* is inserted as a new action in the domain. This action is an alternative action to *turn_to* since it achieves a similar state to the *turn_to* action but with different resources (a battery). Furthermore, the cost of this type of energy is cheaper than fuel. But, it can perform fewer turning activities compared to the action that uses fuel. Suppose the metric function for the problem is changed to minimise *total_cost*. This metric function is then increased according to the numeric positive effect for both actions *turn_to* and *turn_to_battery_power*. In this case, competitor plans could be developed in which one uses more actions, but has lower cost, than the other. This is because the execution of the *turn_to* action might decrease the plan length but increase the metric function. In contrast, performing the *turn_to_battery_power* action can decrease the metric function but increase the plan length. In other words, a conflict presents between obtaining the minimum value of plan length and plan cost. In this domain, the attempt to minimise the plan length does not guarantee an optimal value of the metric function or the plan cost. Therefore, this domain is considered as *semi- straightforward*. The solution development for this problem can be considered computationally hard due to the choices of resources that give different values to the metric function. The conflicting factors between the optimal value of the plan length and metric function in the solution development in such a domain encoding is interesting and it is being investigated in this thesis.

Assume the metric function is to minimise the *fuel-used*. The numeric effect of the above *turn_to_battery_power* is changed to reduce the *fuel-used* by 1. This means that adding more *turn_to_battery_power* actions would actually reduce the value of the metric function. In other words, longer plans might have better metric function values than shorter ones and it would never be a good idea to minimise plan length to optimise plan cost. In these circumstances, the metric function is classified as *expressive*. The solution development of this problem is more complex. However, this kind of domain

is not investigate in detail in this thesis and will be considered as the future work.

The simplest solution to the problem of minimising the plan metric is to ignore the plan metric. The generated plan satisfies only logical goals and only addresses plan quality if the plan metric happens to impose a constraint on the actions. Therefore, many numeric planners use plan length to guide the solution development. This approach is widely adopted in the state-of-the-art numeric planners. The discussion on this is covered in Section 2.6. Although this approach give a feasible solution, it does not fully exploit the benefit of the plan metric. Furthermore, the use of the plan metric allows alternative actions with different costs to be encoded in the domain. Thus, measuring plan cost solely in terms of plan length is not appropriate.

2.3 IPC3 Numeric Domains

IPC3 was the first planning competition that benchmarked the numeric domains and its technologies. The following description explains how the numeric extensions in PDDL2.1 are used in the IPC3 domains [38] [46]:-

- **Depot**

  The Depot domain combines Logistics and the well-known Blocks World domain. The numeric variables are used to encode the capacities of trucks as well as to represent the fuel consumption as the truck moves. Besides trucks, the fuel is also consumed when the hoist is used to lift the packages and crates. The packages and crates are associated with weights. The trucks have capacity constraints and the total weight of crates or packages loaded to it must be less than, or equal to, the capacity constraint.

- **Zeno-Travel**

  Zeno-travel is another transportation domain in the IPC. In this domain, the air planes will fly between locations and consume fuel at different rates according to the speed of travel which is fast or slow. A fast movement is only permitted if the number of passengers is below a certain threshold value. A refuel operator can be applied whenever necessary to obtain the fuel level of a plane back to its maximum capacity.

- **Satellite**

  The main tasks of the Satellite domain is to schedule a number of satellite observation activities. The activities comprise turning the satellites to the right targets,

switching the installed instruments to on or off, calibrating the instruments and taking images. The numeric constraints in the domain represent a finite capacity of the data memory and fuel. The fuel consumption of the satellite is determined by the slew time between targets. All images are stored in a limited amount of data memory.

- **Rover**

  In the Rover domain, data collection at waypoints and data transmission to landers activities are planned for a number of rovers. The activities involve navigating the rovers, taking or dropping samples, calibrating cameras and taking images and communicating the data to a lander. These activities decrease the energy of the rover which then can only be recharged at sunny locations.

- **Settlers**

  In general, the Settler domain is about building facilities such as *housing*, *railway tracks*, *wharf*, *docks* and *sawmill* in some locations in an unsettled area. These facilities require the raw materials that have to go through some processes before they become available for building processes. The raw materials such as *timber*, *stone* and *ore* must first be fallen, broken or mined and some of them are transformed into a refined material, like *timber* into *wood* or *coal* and *ore* into *iron*, before it can be further used. The transformation activities are performed in the facilities which have to be built first. For example, the *saw-wood* action which transforms the *timber* to *wood*, has to be done in the *sawmill*. Furthermore, timber is also required in order to build the *sawmill*. The material sometimes have to be transported to locations where the source of raw materials is not available. Therefore, vehicles like *carts*, *train* or *ships* must first be built depending on the availability of the transportation modes in the area.

  In order to model the above activities, numeric fluents are used to construct resources in the actions. Then, the constructed resources are consumed by other actions in the domain. In other words, numeric interactions exist in this domain. For example, the *saw-wood* action increases the amount of *wood* and decreases the amount of *timber* at the respective location by one unit. The *sawmill* action decreases *timber* by two units and increases *sawmill* by one unit. The *sawmill* action has to be performed before the *saw-wood* action since its effect is the precondition of the *saw-wood* action. Transportation of the materials from one location to another also increases and decreases the amount of the transported materials at the respective locations. Building some facilities like *housing* and

iron-work involve more than one type of material like *wood* and *stone*. This task will increase the unit of facilities but decreases the *stone* and *wood* availability at the location. Numeric fluents are intensively encoded in this domain compared to other domains in IPC3. Resources are mainly *consumed* by the actions in Depot, Zeno-Travel, Satellite and Rover. But in Settlers, the resources are *exchanged* from one action to another in which the consumption of one resource also results in producing another new resource.

2.4 Numeric Domains Analysis

The previous section has discussed the usage of numeric fluents in the IPC3 domains. Generally, all the domains exploit the numeric fluents to model resources. The domains however, are also available in a temporal version. But, this analysis is conducted only on the numeric domains. In this section, we would like to examine the metric function encoded in the numeric domains described in the previous section. Almost all of the problem instances created for each domain employ the plan metric. The aim of the analysis is to investigate whether the metric function encoded in the domains can fall into any of the categories defined in Section 2.2.1: *strictly straightforward metric*, *straightforward metric*, *semi-straightforward metric* and *expressive metric*. In short, the solution development for an *semi-straightforward* problem is considered computationally hard since alternative solutions can be produced and the quality is measured by the value of the metric function. Furthermore, the conflict between plan length and plan cost that often exists in the problem, contributes to the complexity of the solution.

The metric function focused on in this analysis only involves numeric fluents though the metric functions used in IPC3 also include time. Basically, the metric function designed for time, usually written as *(minimize: (total-time))*, is used to obtain the minimum time span for all activities in the generated plan. The numeric version of *Zeno-travel* domain also include time, combines with other numeric fluent called *total-fuel-used*. Other numeric version of domains such as *Depot*, *Satellite*, *Rover* and *Zeno-Travel*, are encoded with metric functions that attempt to minimise the numeric fluents such as *fuel-cost*, *fuel-used* and *recharge*.

Generally, a planning problem encoded with a plan metric is defined as *Semi-straightforward* whenever the following features can be identified in the domain:

- more than one action achieving the same goal or sub-goal states, but giving different metric effects through different metric variables

- more than one action achieving the same goal or sub-goal states, but giving different values for the same metric variable
- resource choices in which different resources result in different metric function values.

The above features will be analysed in the IPC3 domains. Figure 2.2 exhibits the possible flow of the action sequences that would be developed in solutions for problems in the *Satellite* domain. The alternative actions that achieve similar states are not captured in the domain. As can be seen from the figure, there is only one flow of action sequences directed to *take_image*, the action that is responsible for achieving the goal state. In this problem, the solutions can be different between one to another in terms of the number of *turn_to* actions, as there is a repeated loop in that action. The metric function in the problem is *minimise fuel_use*. The metric constraints on fuel are encoded as preconditions in the *turn_to* action, the action to be minimised in the solution. Therefore, the metric function is already encoded in the solution development. In other words, the metric function is *strictly straightforward* in this problem. Furthermore, the resource in the domain in not renewable and does not have any constraint which means optimisation is not necessary. If other images are added to the goals, the generated plan would contain repeated copies of the same plan.

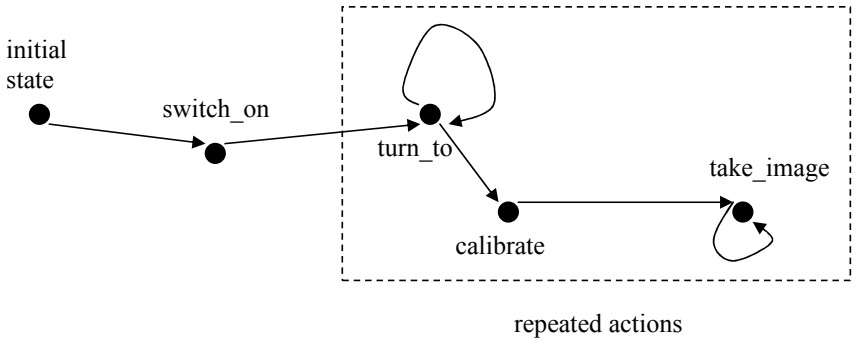

Figure 2.2: Action Sequences in Satellite Domain

A very similar pattern of action sequences is also discovered in the *Rovers* domain. See Figure 2.3. In this domain, the metric function is to minimise the *recharge* value. The *recharge* value increases 1 unit every time the *recharge* action is executed. Therefore, in order to have the minimum value of the metric function, the number of *recharge* actions should be as small as possible. Similar to the *Satellite* domain, the

optimum value of metric function is obtained simultaneously whenever minimising the plan length in the solution development. Following this, the domain is classified as *strictly straightforward*.

Actions with different levels of consumption of the resources seem to be presented in the *Zeno-travel* domain. There are two actions affecting the metric function in the domain: the *zoom* and *fly* actions. Both actions achieve the same sub-goal which is to transfer a person from one to another location, but the actions require different amounts of fuel. However, these actions are not considered alternative to each other as an extra constraint, *(≤ (onboard ?a) (zoom-limit ?a)))*, is imposed on the *zoom* before it can be executed at any point in the plan. The metric function in this domain is classified as *straightforward* since reducing the number of actions does not guarantee giving an optimal value of the metric function.

The *Settlers* domain exhibits interesting numeric interactions as explained in the previous section. In contrast to the above domains, resources defined in *Settlers* are not only consumed but also produced by the actions in the domain. There are three metric function variables included in the domain problems: *labour*, *pollution* and *resource-use*. These metric function variables are defined as a numeric positive effect in the actions rather than as numeric conditions as in the previous domains. This domain still poses a challenge to the numeric planners to solve the problems mainly due to its numeric interactions behavior. Besides its numeric interactions feature, the metric function in this domain is defined according to Definition 3: it is *semi-straightforward*. In this problem, there is always a choice to be made when such resources are required. The resources can either be produced at the target location or transported from another location. These two strategies of acquiring resources will produce plans that have different metric effects and plan lengths.

In sum, although all the problems in the metric domains in IPC3 incorporate a metric function, only *Settlers* exhibits the *semi-straightforward* feature. In other domains the metric plan quality solution is constructed along with the minimisation of the actions in the plan. In other words, the optimal value of the metric function is generally obtained by minimising the plan length. The alternative actions that have potential to construct several alternative solutions, each with different values of the metric function, are not encoded in the domains.

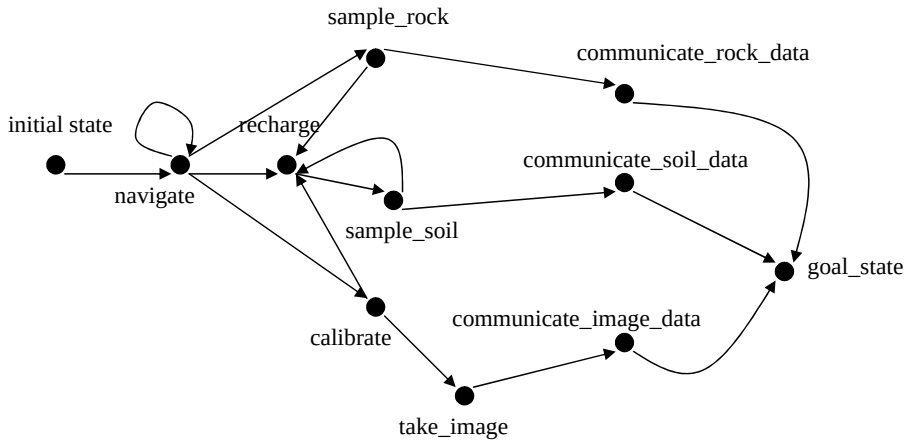

Figure 2.3: Action Sequences in Rover Domain

2.5 Approaches in Planning Systems

Among the techniques that increases the efficiency of planning systems is the *planning graph*. The first work based on this is GraphPlan[4]. In Graphplan, a compact structure called a planning graph is constructed and annotated. The planning graph is a levelled graph that alternates between proposition levels and action levels. It is also a directed graph in which edges represent the relations between actions and propositions. The propositions at layer0 represent the current state, actions at layer0 are all applicable actions, whose add lists made up the proposition layer 1, and so until either the goal or a fix point is reached [37].

Another significant approach to the planning technology is heuristic search as proposed by Bonet and Geffner[5, 6]. The heuristic search algorithm performs forward search from initial state to a goal state using a heuristic function. The heuristic function, $h_G(s)$ is an estimate of the number of steps needed to go from a state s to a state $s^{'}$ that satisfies the goal G. It relaxes or simplifies the planning task by ignoring the *delete list*. HSP[6] (Heuristic Search Planner) and FF (Fast Forward) [40] both handle STRIPS and perform heuristically guided forward search from the initial state to the goal state. But, the FF heuristic technique is relaxed GraphPlan. A heuristic is calculated based on relaxed planning graph in which the graph is constructed similar to GraphPlan. Other planners that adopted this heuristic are MetricFF [38] and SGPlan [8].

2.6 Numeric Planners

This section discusses approaches adopted in some of the state-of-the-art numeric planners in solving numeric domains. The state-of-the-art numeric planners included in this discussion are Metric-FF, LPG and MIPS. The first two are Graphplan based planners, whilst the last one is based on an efficient model-checking approach combined with the use of the relaxed planning graph in the heuristic construction.

- **Metric-FF**

  One of the top performing numeric planners in IPC3 was Metric-FF [38]. Metric-FF extends the capability of Fast Forward (FF) planner[40], which is its predecessor, to handle numeric planning. Many ideas developed in its predecessor are also inherited by Metric-FF. It uses a two-step process in building its heuristic similar to FF. The first step deals with the construction of a relaxed planning graph and the relaxed plan is extracted at the second step provided that the graph succeeds in reaching the goals [38]. During the construction of the relaxed planning graph, Metric-FF extends relaxation of the proposition applied in FF to the numeric facts. This means both negative propositional delete effects and decreasing numeric effects are ignored in Metric-FF. The numeric variables are also estimated during this stage using the bounds propagation technique. This technique estimates the upper and lower bounds value for each numeric variable by summing the action effects.

  The relaxed plan is extracted through a backwards loop from the goal layer to the initial layer. Basically, at the goal $layer_n$, actions to support the respective goal set will be selected. These action preconditions become sub-goals at $layer_{n-1}$ and have to be satisfied by actions at $layer_{n-1}$. This backward loop is repeated until the first layer is reached. The selected actions are used to form the relaxed plan. The heuristic value is based on the number of actions in the relaxed plan. Chapter 5 gives a detailed explanation of how Metric-FF develops its heuristic estimate. It also discusses the drawback of negative numeric effects. The heuristic in Metric-FF, in summary, attempts to minimise the plan length of the solution. Since it tries to minimise the number of actions in the plan, the heuristic only includes the numeric precondition variable of the actions. Therefore, the quality solution can be obtained only if the plan metric variables is encoded in action precondition in the domain. For example, plan metric variables have been encoded as action preconditions in the *Satellite*, *Depot*, *Zeno-Travel* and *Rover* domains in IPC3.

- **LPG**

  Local search Planning Graphs (LPG) is another numeric planner based on the planning graph. Instead of building a single planning graph, it uses a sub-graph called a Numerical Action graph (NA-graph) to represent a partial plan [33]. The NA-graph is a directed acyclic levelled graph alternating a fact level and an action level. The fact levels contain propositional nodes and numerical nodes. Each action level contains one *action node* labelled with the name of a domain action. Any action a at level l is connected by *precondition edges* and *effect edges*. The precondition edges are the propositional/numeric nodes at level l whilst the effect edges are the propositional/numerical nodes at level $l+1$. The problem goals are the precondition of the action, a_{end} and the effect of the action, a_{start}, contains the initial facts of the problem. A numerical action graph is defined as inconsistent if there is an action with the unsupported precondition, θ. Meanwhile, the solution graph is the NA-graph without *inconsistencies*.

  To search for the solution graph, LPG uses a local search known as Walkplan. Generally, the local search selects an unsupported precondition, θ, in the current NA-graph for a particular numerical state, A, and identifies its neighborhood. The definition of the numerical state A is given in [33]. The search neighborhood can be derived by removing the action, a with precondition θ or adding an action node, a, that can resolve θ. The addition/removal of an action node, a, at level l can affect the values associated with the numerical fluents at the next level. An *action evaluation function*, E, estimates the cost of adding or removing the action node a. Both the insert and remove part of E consists of two weighted terms: the search cost and the execution cost. The first parameter estimates the number of steps needed to reach a solution whilst the second parameter estimates the increase in the plan execution cost or the quality of the current partial plan. The parameter weights are dynamically evaluated during search using discrete Lagrange multipliers. The relaxed plan, π, consists of an estimated minimal set of actions to achieve θ. LPG also searches for the minimum number of actions that achieves a particular numeric goal.

  LPG improves the plan quality by producing a sequence of plans. The first generated plan is used to initialise a new search for the next plan. Some *inconsistencies* are enforced in the NA-graph representing the best plan, Π, so far, in order to start a new search. In searching a new solution plan, LPG will remove some actions from Π. It prefers those actions with the high execution cost if the plan metric expressions require minimising a numerical expression. The plan cost is

not directly taken from the plan metric. LPG uses a random search algorithm that chooses the best successor with probability $1 - p$.

- **MIPS**

  Model Checking Integrated Planning System (MIPS) [14] [17] is another distinguished planner in IPC3. This planner is based on symbolic model checking technologies. Its general architecture is divided into four parts: precompilation, heuristic, search algorithm and post compilation. However, its heuristic development is influenced by FF for the propositional part [40] and Metric-FF [38] for the numeric part. Besides a relaxed planning graph heuristic, MIPS also employs the pattern database heuristic [15] for the propositional parts. The numeric heuristic is described by the number of steps required to achieve the numerical goal independent from the propositional. The relaxed planning graph for numeric facts is built by computing a fixed point of a state vector restricted to monotonically increasing propositional and numerical variables. Each numeric value is in the form of the vector interval (min,max). Starting with the initial states values, these values will be updated in the while-loop until it reaches a relaxed fix point value. The fix point for the variable domains is computed by considering the numerical effect in the operator set only. In each iteration, every operator is tested for applicability by checking all numeric preconditions with the current vector of the interval(min,max). Both propositional and numeric heuristics are combined in a unified plan graph. In the forward phase the effects are applied to generate the layered structure of the relaxed planning graph. In the backward phase, to obtain the relaxed plan the preconditions are used for propagation.

In summary, the above numeric planners do not include the constructed plan metric variables in developing the solution. Generally, the heuristic built in these planners aim to minimise the number of actions in the plan.

2.7 Planning and Scheduling Problem

Planning and scheduling are closely related problems. Planning is defined as a process of finding *what* actions to apply by reasoning about the consequence of acting in order to choose among a set of choices of actions[11]. The input to the planning process entails a set of initial states, goal states and possible actions. Whereas, the output or solution to a planning problem consists of one or more courses of action that lead from the initial states to the goal states. Most planning problems are computationally hard in the sense that the number of possible courses of actions is exponential [11] and the

complexity of the solution development is categorised as PSPACE-complete [7]. The complexity results from cascading sets of action choices that interact in complex ways as well as the fact that plan length is unknown in advance.

On the other hand, scheduling, as defined in many Operational Research books is a problem of assigning limited resources to the actions over time to optimise one or more objectives [57]. In this definition, the resources become the core part of the problem. Planning and Scheduling are mainly distinguished by the facts that scheduling problems are not enriched with action choices as in planning. Therefore, in constructing the solution, scheduling algorithms are not concerned with making the *what* actions choices but rather with figuring out *when* and *with what* resources the predefined sequence of actions should be carried out. For example in an established job-shop scheduling problem, there is a set of jobs where each job has a set of ordered tasks with specified duration and set of machines. Each machine is capable of carrying out a subset of the set of all tasks [11]. Sometimes action choices can exist in this problem, for example choices about resources and perhaps the process alternatives. But, these choices are highly constrained. Some tasks may be optional and some tasks may allow simple process alternatives.

The resources and action choices continuum in planning and scheduling problems can be viewed as in Figure 2.4. Many real world applications are generally to be found in the middle of the continuum where the problems have almost equal density of the resources and actions consequence reasoning. The examples of real world applications that fall in this category are planning and control of supply and distribution logistics, project management for product introduction, systems engineering construction, process flow of assembly, and so on [13]. Realising the importance of resources in planning, attempts to treat resources in planning have been made and among the earliest development made are in [13] [44] [43]. These attempts also mean to bring scheduling elements into planning. In contrast to scheduling, the resources are not explicitly modelled in planning problems. They can be treated in different ways. An object in planning can act both as a resource and as a consumer of resources [59]. For example, the ZenoTravel domain in IPC3, the *plane* is a resource when the goal is to transport people from one city to another but it is also a consumer of resources if it itself must be at the certain location. The numeric domains benchmarked in IPC3, have witnessed that another significant effort has been made in the planning community to bring resources into the planning problems. However, these benchmarked domains are still lacking real resource choices; the important element of scheduling. Therefore, the domains in general do not really exhibit a strong element of scheduling.

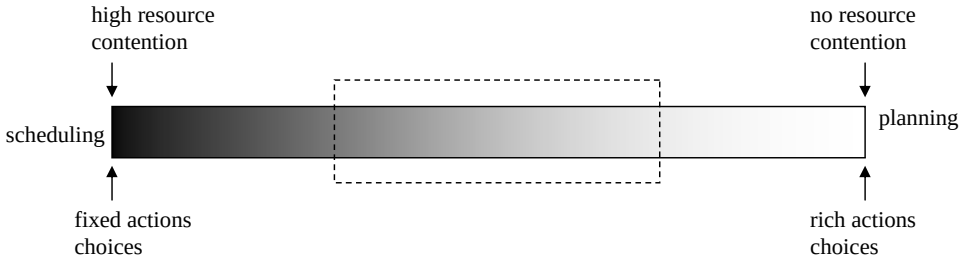

Figure 2.4: Resource and Action Choices Continuum in Planning and Scheduling

Including resources in planning results in adding more constraints to the planners since they can influence the shape and structure of the plan. The resource can impose constraints on the number of actions that can be executed in parallel. This happens if the number of resources available at each point in time throughout the plan is limited. This constraint is known as *resource-width*[47]. Furthermore, resources also can impose constraints on the number of actions that can be freely executed during the lifetime of the plan. This constraint arises from the limited quantity of resources throughout the plan and it is also known as *resource-length*[47]. However, the resource-width constraint is irrelevant to planners that produce totally ordered sequences of actions[47]. Following this, apart from maintaining the ordering relationship among actions in developing a solution, the planners have to ensure there is no conflict in resources, particularly, in the limited resources [59]. Importantly, once resources are introduced into the planning problem, the cost and consumption of resources become new attributes to the plan value. The plan evaluation is far more complex as different resources might have different values [47].

Combining scheduling into planning means that the solution of the problem now requires the construction of *what action* choices with *what resource* choices. The resource constraints and choices, together with the choices of actions lead to a number of feasible action sequences. Each of these action sequence produces a different objective value. Furthermore, the resource constraint can impose a new resource constraint if it is further being used to create a new resource object. The flow of possible action sequences in planning and scheduling can be illustrated in Figure 2.5. The branches in a state implies the alternative actions available in achieving the next state. For example, the set of feasible solutions nodes are $(0, 1, 3, 5, 7, 8)$, $(0, 1, 2, 4, 6, 8)$ and $(0, 1, 3, 4, 6, 8)$. These solutions give different values to the plan cost. Suppose the plan cost of each set are 5, 6 and 8 respectively. The loop at node 4 in the diagram indicates the repetition

of the unknown action sequences, resulting from the new constraint imposed by the current state in the plan. For example, if a set of actions $(0, 1, 2, 4, 6, 8)$ is selected, the next set of actions would be $(0, 1, 2, 4_1, 4_2, 4_3, 6, 8)$ depending on the state resulting from the effect of the action in node 4. Without the loop, the choices of action sequences in the Figure 2.5 is known and exhibits more scheduling features rather than planning. This is due to the fixed action choices of the action sequences. The number of times around the loop cannot be determined in advance, so the action sequences cannot be predicted from the beginning of the process. Such problem features exist in the middle of the continuum from planning to scheduling.

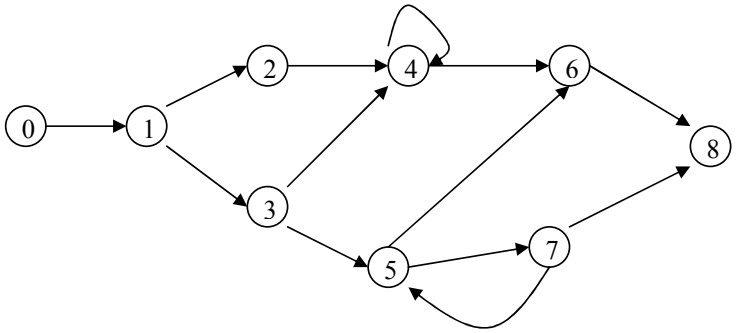

Figure 2.5: Metric Planning Action Choices Flows

2.7.1 Integrating Planning and Scheduling

There are many approaches that have been taken by researchers in their attempts to develop solutions for the integrated planning and scheduling problem, precisely in handling resources in planning. One clever approach is to tackle the two processes independently [27]. This approach is applied in *RealPlan* [58]. In this approach, the decision about *what* usually comes first and is later followed by *when* and *with what* resources in generating the plan. The main advantages of this approach is the possibility of obtaining an abstract plan, which is independent from the resources and it can be reusable. But, the solution is not globally optimal and there is the possibility of excessive interactions between the planning and scheduling processes [27]. The **IPP** [43] planner supports the ADL language, and incorporates resources in the graphplan framework where the resource is strictly action-centred. This means, specific requirement of the resources and the resources effect are determined every time an action is instantiated.

Some researchers exploit scheduling tools, particularly linear programming, in the development of their solutions. **LPSAT** is an example[63]. It converts resource plan-

ning into the Satisfiability problem (SAT) and uses linear programming as a subsolver for the numeric part. Another planner that handles resources constraints through mathematical equations is the ZENO planner [52]. The O-Plan and IxTeT planners solve resource conflicts using techniques similar to scheduling methods. O-Plan[13] uses optimistic and pessimistic resource profiles to detect and solve potential resource conflicts. IxTeT [44] uses a graph search algorithm to identify the minimum critical sets of actions with conflicts and plots disjunctions of ordering constraints when solving a conflict[47]. The systems described in the previous, however, do not address the problem of more complex plan evaluation. Generally, in the above works, the resources are only used as constraints, not as measures to be optimised.

2.8 Multi-objective Planning

Traditionally, plan evaluation is purely based on the plan length and this approach has been adopted in many domain independent planners. The discussion on this was covered in Section 2.4. However, in real world problems, other criteria such as resource consumption or profit also play an important role in the solution. This is also has been extensively discussed in the previous section. In planning with resources, particularly, the plan produced should also minimise the cost of the resources. Therefore, the generated plan must be evaluated based on more than one criterion. This approach can be called multi-objective planning in which both plan length and plan cost are taken into consideration in the plan evaluation. Amongst the works in the literature that claimed to use a multi-objective optimisation approach in their heuristic include MO-GRT[54] and TPSYS[28].

MO-GRT [54] is a domain independent heuristic state-space planner that works under the STRIPS framework. It extends the heuristic in its predecessor, called GRT[53]. In GRT, the heuristic function estimates the distance, in terms of the number of actions, between any state and goal state. The aim of the heuristic is to minimise the plan length, the common plan evaluation adopted in many domain independent planners. Besides the plan length, MO-GRT's heuristic function deals with the multi-objective criteria by including the information on resources. It is considered as the first attempt to apply a multi-objective evaluation technique in the area of domain independent planning. The multi-objective heuristic is accomplished by assigning each fact, p, with a set of *cost vectors* in the form, $< length, C_1, C_2, \ldots C_N >$. This cost vector estimates the cost of various alternative paths that achieve p from the goal and N is the number of criteria that are being considered in the plan. MO-GRT adopts the Weighted Average Sum method (WAS) in which weights have to be assigned to each criterion to represent

their relative preferences to the user or evaluator. This results in a basic ranking among the alternative solutions. Besides the weight, a scale, the range of allowable values also assigned to each basic criterion. The weight and scale are defined by the user. The cost vector is evaluated based on these weights and scales and generally the action with the best cost vector is chosen in the solution development.

TPSYS [29] is a graphplan-based temporal planner that manages duration action as proposed in level 3 of PDDL2.1. This work later is extended in[28] by including numeric fluents. The work is claimed to be a form of multi-objective planning because the solution development includes both metric and time variables. These two parameters however, are considered separately during the solution construction. The solution development in this planner is divided into two main stages. The first stage called spike construction in which each proposition/action is associated with a cost vector. This vector includes the variable and its estimated minimum and maximum values to achieve in the proposition/action. Initially, for each proposition, the numeric variables is initialised with the values indicated in the initial state. The variable values are incrementally updated through the spike construction together with the instantiation of the actions. The second stage is called the search for a plan and is divided in two stages. First, generate the initial relaxed plan which consists of actions that are evaluated and selected according to the estimation of their numeric variables. At the second stage, the execution time of each action will be allocated to the plan generated in stage 1. The multi-objective optimisation in this case concerns allocating time to the action within the estimated value of the numeric variable.

2.9 Multi-objective Functions in Linear Programming

Linear Programming (LP) is an optimisation tool that is widely used in the operational research community to solve optimisation problems. The classical LP model consists of a linear objective function and a set of linear constraints. The community, however, has recognised that many problems addressed by single objective optimisation can be viewed as multi-objective in nature. Therefore, the community has started to consider the multi-objective problem and has provided a Multi-objective Mathematical (MOMP) framework that considers multiple objectives explicitly and simultaneously[20]. The multi-objective mathematical programming problem is defined and stated according to[45],

Definition 5 (Multi-objective Mathematical Programming) *The multiple objective mathematical programming generally consists of more than one objective. Without loss of generality, all objectives are assumed being maximised, therefore it can be stated as the following :*

$max\ \bar{Z}(\bar{x}) = [Z_1(\bar{x}), Z_2(\bar{x}), \ldots, Z_p(\bar{x})]$

subject to constraint $\bar{x} \in X$ *where:* $Z_1(\bar{x})$ *,* $Z_2(\bar{x})$*, ...,* $Z_p(\bar{x})$ *are the problem's* $p \geq 2$ *objective functions*

$\bar{x} = (x_1, x_2, \ldots, x_n)$ *are the problem decision variables and* X *is the explicitly defined feasible region (Note: minimisation of the objective functions is the negation of the maximisation of the objective function)*

Example of multi-objective mathematical problem according the above definition can be stated as follow,

$$\begin{aligned}
&\max\ Z_1(x) = x_1 + x_2^2,\\
&\quad Z_2(x) = x_1^2 + x_2,\\
&\text{subject to :}\\
&\quad g_1(x) = 12 - x_1 - x_2 \geq 0,\\
&\quad g_2(x) = x_1^2 + 10x_1 - x_2^2 + 16x_2 - 80 \geq 0
\end{aligned}$$

If all the objective functions and constraints are linear the mathematical model is called Multi-objective Linear Programming (MOLP). This MOLP is a problem that minimises several linear objective functions simultaneously subject to a set of linear constraints and a set of linear decision variables [64]. Generally, however, the optimisation of MOLP problems rarely results in a unique optimal solution due to the presence of conflicting objectives. It is common in MOLP problem, that at least two of the p objective functions are conflicting in nature. For example, minimising the process time can maximise the cost of a machine due to the increase in the machine capacity. According to [50], the term *optimise* means to find all the values of the objective function acceptable to the decision maker. The optimisation of the MOLP problem consists of more than one solution and is referred to as the Pareto-optimal set. Each point is optimal in the sense that no improvement can be achieved in one cost vector component that does not lead to degradation in at least one of the remaining components [22]. The Pareto-optimal front gives a set of solutions called the *non-dominated* solution.

Definition 6 (Non-dominated Solution) *$\bar{x}^* \in X$ is a non-dominated solution if and only if there does not exist $\bar{x} \in X$ such that $Z_i(\bar{x}) \geq Z_i(\bar{x}^*)\ \forall i = 1, 2, \ldots, p$ and $Z_i(\bar{x}) > Z_i(\bar{x}^*)$ for at least one $i = 1, 2, \ldots, p$*

One of the principles to find the *non-dominated solutions*, which is employed in many heuristic techniques, is *scalarisation*. This technique basically transforms the multi-objective problem to a single objective problem that is solved repeatedly[19]. The classical method to obtain the non-dominated solution in the scalarising function is called *weighted-sum*. Basically, in this method different weights are given to different objective functions [9].

Definition 7 (Weighted-Sum Technique) *The weighted-sum techniques is used to specify a scalarising function by giving weights to the objective functions in the form: Minimise $\sum_{i=1}^{k} w_i Z_i(\overline{x})$ where $w_i \geq 0$ are weighting coefficients representing the relative importance of the objectives. It is usually assumed that $\sum_{i=1}^{k} w_i = 1$*

The set of Pareto-optimal solutions can be obtained by systematically changing the weight among the objective functions. Therefore, the solution can vary as the weighting coefficients change. Generally, the designer chooses the weight based on intuition since there is no proper guideline to be used in choosing the weight. The weighting factors do not proportionally reflect the relative importance of the objective, but it is used to locate points in the Pareto set, If the weight, w_i closely reflects the importance of the objective, the function can be transformed into the following [9],

Minimise $s \sum_{i=1}^{k} w_i Z_i(\overline{x}) c_i$
where c_i are constant multipliers that will scale the objective properly.

However, to obtain the best results, the vector function must be normalised first[9]. Generally, the weighted-sum technique is computationally efficient and generates a strongly non-dominated solution which can provide initial answers to other problems. However, the appropriate weights can be difficult to determine particularly if the detailed information about the problem is not available[9].

Linear programming (LP) has also been adopted as the solution technique in planning as in [42, 60, 62]. But these works only use a single objective function and LP is generally used to obtain the heuristic that optimises the plan length. This means the quality of the solution is determined by the plan length. In planning problems that can be defined according to *semi-straightforward metric function* problem class, however, the plan evaluation is based on multi-objective criteria: plan length and plan cost.

Therefore, this thesis attempts to apply the multi-objective function in the extended heuristic function to deal with these parameters. The detail about the implementation of the multi-objective criterion in the developed heuristic is discussed in Chapter 5.

2.10 Conclusion

This chapter has provided an overview of the current knowledge of modelling resource optimisation, particularly, in the field of numeric planning. There are various ways to include resources in planning, and different ways of defining optimal solutions. However, most of the benchmarked numeric domains are found to have adopted a common approach in which the optimisation of the resources is straightforwardly obtained whilst minimising the number of actions in the constructed solution. Many numeric planners that handle PDDL2.1 have also developed heuristics that aim to minimise the value of the plan length. In other words, the most common plan evaluation is still based on a single perspective which is the number of actions in a plan. The plan metric is used to evaluate the solution *post hoc*. This chapter also discussed the planners that claim to have used multi-objective approaches in their solution construction. These planners, however, include time as another objective besides the plan length. Few researchers have considered using the plan metric to guide the solution development. An overview of Multi-objective Linear programming, the optimisation tool that uses more than one objective function, and its solution technique is given in this chapter. This optimisation tool is implemented in the extended planner described in this thesis, in order to obtain the heuristic that balances plan length and plan cost in the solution.

Chapter 3

Domain-Specific Metric

3.1 Introduction

This chapter describes and constructs a new class of metric planning problems called *domain-specific metric* planning. The domain-specific metric class is the problem class for which the extended planner developed in the thesis is designed. The metric functions defined in such problems are *semi-straightforward*. The general structure of the domain includes numeric interactions as well as resource choices that give different values with respect to the metric function. Therefore, various solutions, each with different metric function values can be developed. This structure often exists in many real world applications, however, it has been largely ignored by the planning community. The challenge posed by this class is to optimise values of the metric function. Furthermore, the solution development exhibits trade-offs between plan length and plan cost. This thesis attempts to construct a heuristic function that can obtain a better balance between plan metric and plan length values. The metric function of the problem is included in the heuristic to guide search for the action with the minimum resource cost. Considering both plan length and cost simultaneously in the heuristic can be viewed as a multi-objective planning problem in the sense that more than one criterion will be used to evaluate the plan.

Besides describing the new class of problems, this chapter also reviews some benchmarked numeric domains. Numeric fluents and metric functions were also encoded in the metric planning problems benchmarked in the previous competition series[12, 39, 46]. However, the focus given in modelling the metric functions and numeric fluents differs from one competition to another. This chapter examines the characteristic of the benchmark problems in the IPCs series so far, particularly those involving numeric fluents.

3.2 Characteristics of the Planning Domains in IPCs Series

In the International planning competitions series, in short IPCs, a set of benchmark problems is provided for others to use to compare their system to the state-of-art planning technologies [49]. The first series of the competition was held in 1998 in conjunction with the International Conference on AI Planning and Scheduling (AIPS) in Pittsburgh, Pennsylvania. The Planning Domain Definition Language (PDDL) was created by the AIPS committee and first used as the official language in the first competition [39]. The other ultimate goals of the IPCs series include to provide an indication of overall progress in the planning technologies, encourage the development of more realistic problems and to develop and improve the PDDL language itself [1, 24, 34, 39, 49].

The focus of the planning community on planning technology development and characteristics of the problems being solved, varies from one competition to another. The first and second competitions (IPC1 and IPC2) were involved with both STRIPS [21] and ADL [51] domains and problems. The first competition to deal with numeric fluents was IPC3. This competition also focused on temporal planning. PDDL2.1 level 3, the official language for the competition, is augmented with durative actions [24]. The discussion of the numeric domains involved in IPC3 was given in the previous chapter. In summary, the *Settlers* domain was found to demonstrate interesting numeric interactions and to exhibit *semi-straightforward* metric features. The numeric interactions were then later identified by Coles et all as *producer* and *consumer* actions [10]. According to Coles et al., a *producer* is an action that causes an increase in the numeric resource quantity whilst a *consumer* is an action that causes a decrease in the numeric resource quantity. The rest of the numeric problems in IPC3 are strictly straightforward, or straightforward, as defined in Chapter 2.

The IPC4 competition continued to use PDDL2.1 and made some extensions to the language resulting in PDDL2.2. PDDL2.2 is PDDL2.1 extended with derived predicates and timed initial literals [39]. For the first time the competition was separated into a deterministic track, which can be considered a continuation from the previous competitions, and a probabilistic track. IPC4 also attempted to pose more realistic problems. There were five new domains which include *Airport*, *Pipesworld*, *Promela*, *PSR* and *UMTS*. A detailed description of these domains is given in [36]. However, the majority of the domains, except *Promela*, stressed the role of derived predicates and timed initial literals. The *Promela* domain models deadlock detection in the commu-

nication protocols formulated in the Promela language [16]. Nevertheless, the usage of the numeric fluent in this domain is simple, and makes no advancement over the numeric domains in IPC3. In fact, it excludes the plan metric.

The following competition, IPC5, made an effort to draw back the attention of the community to the important issue of plan quality. It aims for a better characterisation of the meaning of plan quality [1]. Along with this competition, the new language PDDL 3.0 which allows the expression of strong and soft constraints on the plan trajectory, as well as strong and soft problems goals, was developed [31]. In general, the strong trajectory constraints and goals must be satisfied by any valid plan, while soft trajectory constraints and goals (called preferences) express desired constraints and goals, which do not necessarily have to be achieved. With these constraints, the plan metric can consist of a function whereby it minimises the violations of the constraints.

The benchmark problems involved in this event were *Travelling Purchaser*, *Openstacks*, *Storage*, *Trucks*, *Pathway* and *Extended Rovers* domains [12]. These domains were presented in propositional, metric-time and simple and complex preferences versions [12]. The plan metrics encoded in the metric-time versions for all domains, generally involve numeric fluents and time. Generally, the numeric fluent variables are also encoded as resource preconditions similar to numeric domains in IPC3 except for the *Pathway* domain. This domain not only inherits the numeric interaction features as demonstrated by the *Settlers* domain in IPC3, but also includes another interesting feature which can contribute to a few possible solutions that have different completion times. However, the choices of the processing times are not provided in all problem instances.

3.3 New Domain Class

A new class of metric problems called a *domain-specific metric class* is constructed in this thesis. Domain-specific metrics problems can be defined as planning problems characterised both by the availability of metric resources, and the presence of a plan metric that refers to the metric variables in the particular domain. Figure 3.1 illustrates an example of a problem created for the *Bread* domain and it is used in the experiment conducted in Chapter 6. The complete definition of the *Bread* domain is given in Appendix C.

The metric function *(:metric minimize (+ (+ (* 4(labour)) (*3(pollution))) (*0(energy))))* stated in the problem is domain-specific because *labour*, *pollution* and *energy* are the numeric variables defined within the domain and manipulated by the actions

```
(define (problem prob1)
(:domain bread)

(:objects
kitchen0 - kitchen
machine0 - machine
)

(:init
(ready-to-use machine0)
(=(has-flour kitchen0)15)
(=(ready-mix kitchen0)4)
(=(ready-dough kitchen0)0)
(=(loaf-bread kitchen0)0)
(=(breakfast-bun kitchen0)0)
(=(cooked-bun kitchen0)0)
(=(cooked-bread kitchen0)4)
(=(energy)0)
(=(pollution)0)
(=(labour)0)

)
(:goal (and
 (>=(cooked-bun kitchen0)10)
 (>=(cooked-bread kitchen0)10)
 (>=(breakfast-bun kitchen0)5)
)
)
(:metric minimize (+ (+ (* 4(labour)) (*3(pollution))) (*0(energy))))

)
```

Figure 3.1: Example of metric function in Domain-specific metric problem

in the domain as can be seen in Appendix C. In other words, this metric function is specific to this domain and cannot be applied to other domains. This is in contrast to the generic metric function, such as *(total-time)*, which can be used to all temporal domains to obtain the temporal span of the entire plan. In the domain-specific metric, a set of possible solutions which differ in terms of the plan metric value can be constructed for the same problem. For example, the plan metric in Figure 3.1 above is changed to *(:metric minimize (+ (+ (* 0(labour)) (*3(pollution))) (*0(energy))))*. A different plan would expect to be generated as the solution. In order to achieve this, actions with resource choices must be encoded in the domain. Furthermore, for the

numeric interactions to be interesting the domain must feature *producer* and *consumer* actions. These actions produce and consume the numeric resources in the domain. Furthermore, the possible solutions also differ in plan length. Importantly, the nature of the solution development is conflicting between the plan length and plan cost. Quality solutions seek to minimise the plan metric while also minimising the plan length.

Undoubtedly, the conflicting factors make this problem class more complex. Feasible solutions do not guarantee the optimal plan cost. The conflicting factors also relate to the nature of the real world problem. The cheaper resource might have small capacity so that it can only handle a few jobs. Meanwhile, the expensive resource might have a larger capacity and be able to perform more jobs. More processes or steps are required in a plan that chooses the action that uses the small capacity resource compared to the plan that chooses actions that use the bigger capacity resource. However, the cost of a plan that includes a small capacity resource is cheaper than the plan that selects the bigger capacity resource. Sometimes the optimal solution can be obtained from the plan that combines both resources in achieving a specific value of the numeric quantity.

The circumstances discussed above are exhibited in the domains designed in the following section. Domains that are categorised as domain-specific metric problems include the *Extended Settler*, *Bread*, *Trader*, *Production* and *Sugar* domains. In general, the *Extended Settler* domain inherits the original *Settlers* domain structure but with more resource choices available at a particular state. The *Trader* domain is a logistic based domain. Whilst, *Bread*, *Production* and *Sugar* are manufacturing based domains. However, the *Sugar* domain also combines a logistics aspect. The following section elaborates in detail the example of the *domain-specific metric class* to put this work in context.

3.3.1 Extended Settler Domain

The original *Settlers* domain, as explained in the previous section, includes *producer* and *consumer* actions. The plan metric variables in the problem instances generally consists of *labour*, *resource-use* and *pollution*. The plan metric serves as a mechanism to obtain a plan cost for a particular developed solution. The domain provides alternative choices of resources so that alternative solutions that differ in terms of the plan cost can be generated.

Since the *Settlers* domain has been categorised as *semi-straightforward* as explained in Chapter 2, it is included as one of domains in the domain-specific metric class. This domain however, is extended in order to enrich the choice of resources, particularly

at certain states. Two new actions are added to the domain. The actions are: *build-house-wood* and *fell-timber-machine*. From now on, the *Settler* domain with the two new actions is referred to as the *Extended Settler* domain. These new actions offer more choices for the planner to reach a particular state with different resources at a particular time. The *build-house-wood* action is an alternative to the existing *build-house* action. Both actions can be selected to achieve the same goal/sub goal, called *housing*. However, these actions use different types and quantities of the materials. The existing action requires only *wood* but the new action requires both *wood* and *stone*. The cost of the plan with the *build-house* action may differ from the plan with the *build-house-wood* action. It uses materials that demand the use of raw materials which can be obtained using different actions. The *wood* is a refined material of the raw material called *timber*. The *timber* is obtained either by the *fell-timber* action or by its alternative action, *fell-timber-machine*. Both actions cause the increase in the *timber* value but by different amounts. The *timber* value is increased by 1 unit by the *fell-timber* action, and by 3 units by the *fell-timber-machine* action. Moreover, these actions affect the different plan metric variables. The *fell-timber* action increases the *labour* metric. Meanwhile the *fell-timber-machine* increases the value of the *pollution* variable.

Choosing the different actions in the plan not only results in different plan metric value but also gives different plan length values. For example, this occurs if a numeric goal requires the $timber \geq 3$. One solution can consist of a single step of the *fell-timber-machine* action. The other solution might include three steps of the *fell-timber* action. Now assume, the goal is $housing\ location0 \geq 1$. The possible action sequences in the generated plan are illustrated in Figure 3.2. Each of the action sequences has a different plan cost. The complete domain encoding for the *Extended Settler* is attached in Appendix A. Table 3.1 shows the profile of the new actions compared to the equivalent actions in the *Settlers* domain in the IPC3 version.

Table 3.1: Extended Settler plan metric profiles

action	plan metric		numeric precondition		numeric effect	
	labour	pollution	wood	stone	housing at location	timber at location
fell-timber	1					1
fell-timber-machine		1				3
build-house			1	1	1	
build-wood-house			2		1	

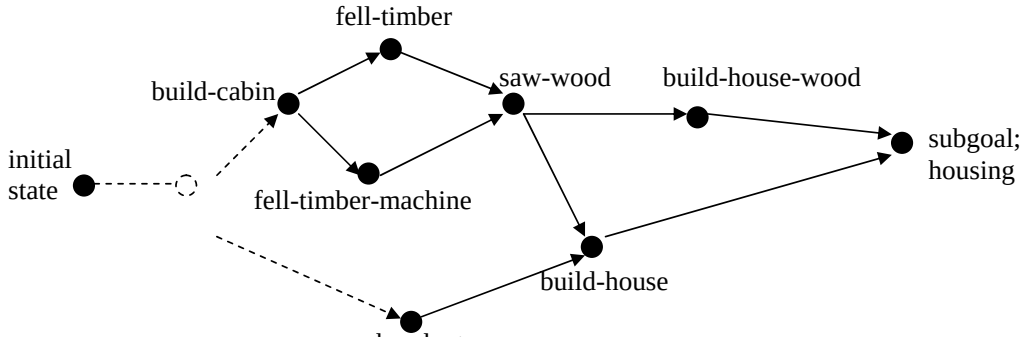

Figure 3.2: Alternative Actions that Achieve Subgoal *housing location0* ≥ 0 in Extended Settler Domain

3.3.2 Trader Domain

This domain is inspired by the Travelling Purchasing Problem (TPP)[12] in IPC5. The *Trader* domain consists of a number of markets each with a set of items with different quantities and prices, similar to the TPP domain. In the original TPP problem, a subset of markets that meet the given demand as well as minimising the combined travel and purchase costs is selected in the solution development. The trader domain basically remains the focus of the solution development as in TPP, but includes choices of resources so that the solution attempts to minimise the plan metric given in the problem instances. The plan metric is attributed by *labour*, *resource*, *expenditure* and *liquidity* variables. In order to make alternative solutions possible in this domain, a few choices of resources are encoded. For example, the *trader* can move the *items* from one location to another using one of these actions: (1)*load-basket* and (2)*donkey-carrier*. The first action may result in an increase to the plan metric *labour*. In contrary, the second action increases the value of the plan metric *resource*. In addition, these two actions are constrained by different values of the capacity. The capacity of the *load-basket* is smaller than the capacity of the *donkey carrier*. If the plan metric attempts to minimise the *labour* cost, the plan will include the *donkey-carrier* action. The *load-basket* action will be considered in the plan, if the plan metric seeks to minimise the *resource* cost. Both plans can have different plan lengths as the maximum capacity of the item that can be loaded at one time is different in these actions. Table 3.2 summarises the important features in each encoded action.

The attribute of the market in the trader domain is also enhanced. The markets not only sell the items but they also buy items from the *trader* and exchange items. Initially in almost all instances created in the domain, the *trader* has some amount of

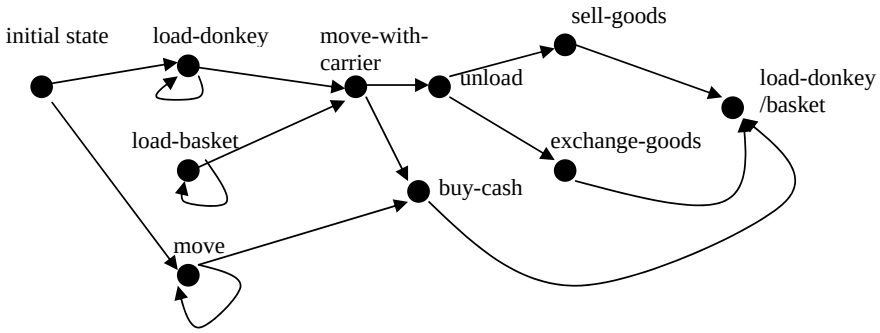

Figure 3.3: General Action Sequences in Trader Domain

cash and items on hand. The goal states include the demand of the other items which are normally different from those that the trader already has on hand. The *trader* can increase the number of the demanded items through either *buy-cash* or *exchange-goods* actions. The *buy-cash* will increase the plan metric *expenditure* whilst the *exchange-goods* increases the value in the plan metric *liquidity*. In order to obtain the required item through the *buy-cash* action, the *trader* must have enough *cash*. In addition, the trader can increase the amount of *cash* on hand by selling the unwanted or excessive items through the *sell* action. Figure 3.3 shows the general flow of the action sequences that may include in the plan solutions. The complete domain encoding is attached in Appendix B.

3.3.3 Bread Domain

The *Bread* domain is a simple example of the manufacturing type domain. This domain is about a process of producing products called *Bread* and *Bun*. Figure 3.4 illustrates the sequence of actions involved in the domain. The major activities in the domain include *preparing mixture*, *kneeding dough*, *making bun or bread* and *baking*. The actions involved in performing these activities exhibit the *producer* and *consumer* features. For example, the actions that perform the kneed activity consume a numeric resource called *ready-mix* which is constructed by the *prepare-mix* action.

Following this, the kneed action also constructs another numeric resource called *ready-dough* which is later being consumed by other actions in the domain. Almost all the *producer* and *consumer* actions in the domain are encoded with their alternative actions. The alternative actions provides a means by which a few alternative solutions

Table 3.2: Trader Domain plan metric profiles

action	plan metric				numeric precondition	numeric effect
	resource	labour	expenditure	liquidity		
ld	1				capacity ≤ 4	item
lb		1			capacity≤ 2	item
bc			1		item	item
eg				1	item	item
sg					cash	cash & item

Notes
ld - load-donkey
lb - load-basket
bc - buy-cash
eg - exchange-goods
sg - selling-goods

relative to the plan metric value are able to be constructed in the problems. The complete domain encoding can be seen in Appendix C.

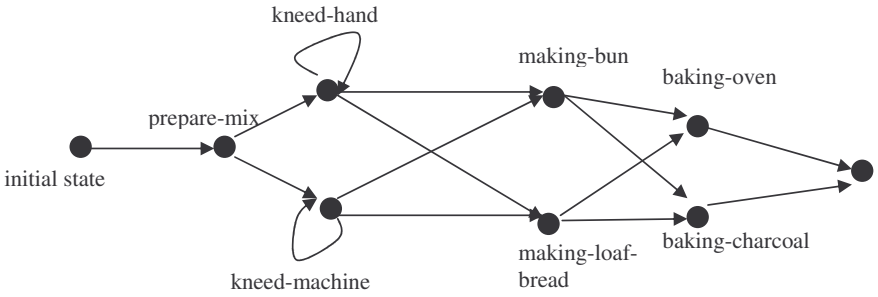

Figure 3.4: General Action Sequences in Bread Domain

The plan metric defined in the domain includes the metric variables *labour*, *energy* and *pollution*. The choice of the *producer* and *consumer* actions appears in the *kneeding dough* process. This process is accomplished either by *kneed-hand* or *kneed-machine* actions. The *kneed-hand* action increases the *ready-dough* by 1 unit and affects the value of the plan metric *labour*. The *kneed-machine* on the other hand, causes a 2 units increase in the value of *ready-dough* and affects the plan metric *energy*. The choices of resources are also exhibited in the baking process. The *baking-oven* and

baking-charcoal actions perform the baking process to produce *loaf-bread* and *breakfast-bun*. The *baking-oven* increases the *energy* variable, meanwhile, *baking-charcoal* action increases the *pollution* value. The profile of the resource choices with regard to the plan metric ! and numeric effects is shown in Table 3.3 and Table 3.4.

Table 3.3: Numeric preconditions and effects in the Bread Domain

action	numeric precondition				numeric effect		
	ready-mix	ready-dough	breakfast-bun	loaf-bread	ready-dough	cooked - bread	cooked bun
kneed-hand	1				1		
kneed-machine	2				2		
baking-oven			10	4		10	4
baking-charcoal			2	2		2	2

Table 3.4: Bread Domain plan metric profiles

action	plan metric		
	labour	pollution	energy
kneed-hand	1		
kneed-machine			1
baking-oven			1
baking-charcoal		1	

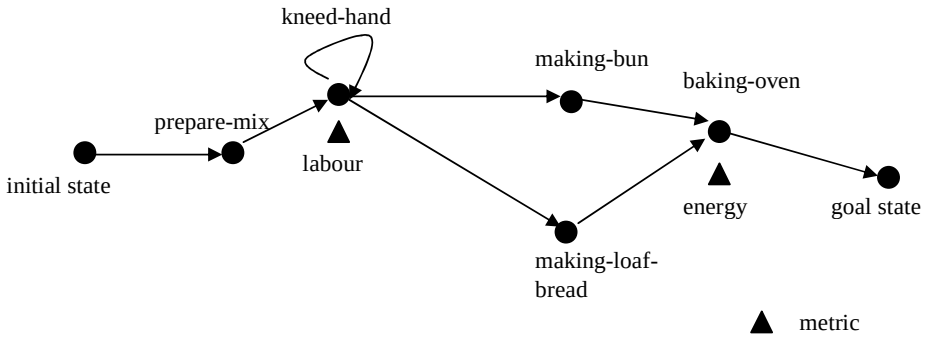

Figure 3.5: Solution with Plan Metric *labour* and *energy* in Bread Domain

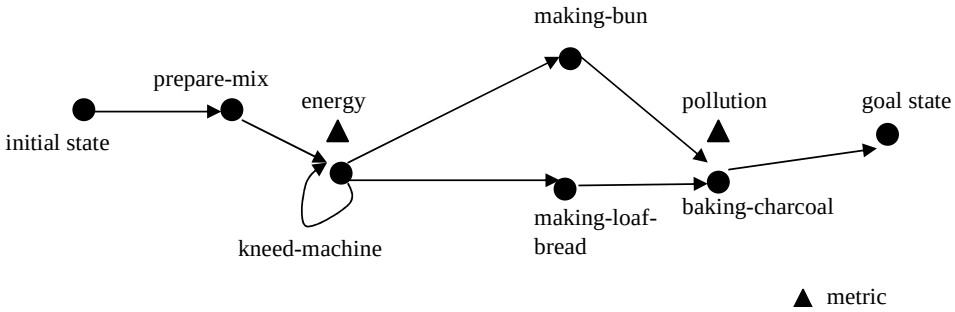

Figure 3.6: Solution with Plan Metric *energy* and *pollution* in Bread Domain

Assume the numeric goal consists of *(≥(cooked-bun)10)* and *(≥(cooked-bread)4)*. Based on the profile given in Table 3.3 and Table 3.4, the alternative action sequences are shown in Figure 3.5 and Figure 3.6. The first feasible solution affects the plan metric *labour* and *energy*. On the other hand, the second possible solution affects the plan metric *energy* and *pollution*.

3.3.4 Production Domain

The *Production* domain is another manufacturing type domain that is constructed in the thesis. It involves more processes, or actions, compared to the *Bread* domain. The *Production* domain is about manufacturing two finished products called *product1* and *product2*. Both products can be constructed using a few different product formulae. Each of the product formulae requires different *parts* or *materials* that have to go through a few different processes. The variety in the product formulae results in a few alternative solutions which differ in terms of the plan length and cost. Every product formula, however, has four stages in common. At the first stage, the *raw material* will be transformed into the *refined material*. Then, the *refined material* is converted into the *shaped material* through the cutting process. The *shaped material* will then go through some processes in order to make the *part*. At the final stage, the parts are assembled according to a formula, to make the products. These stages together with the action sequences involved in the domain is illustrated in Figure 3.7. Similar to the *Bread* domain, this domain exhibits a feature where a resource is constructed by an action and then used in another action in the domain. In other words, the actions in the domain demonstrate the *producer* and *consumer* behavior. The quality of the solution is measured by three plan metric variables declared in the domain: (1) *hazard*, (2) *labour*, and (3) *machine-cost*

Essentially almost all of the stages can be performed using alternatives resources. For example, the *refined material* called *plastic* and *metal* can be obtained from two different actions called; (1) *process-raw-material* and (2) *part-from-recycle*. These actions use different resources in which action (1) uses *chemical* and *iron*, whilst action (2) uses the *recycle parts*. The *recycle part* is a co-product that can be obtained from the making part process. These two actions, however, increase the value of the similar *refined material* with different units. Furthermore, action (1) causes an increase to the plan metric variable called *hazard*. Besides *hazard*, there are another two more plan metric variables being declared in the domain; *labour* and *machine-cost*.

The resource choice occurs again at the making part stage. The quantity of a part called *metal-part*, is increased through two different actions called *painting* and *soak-chemical*. Both actions also consume similar resources, *shaped-metal*, but affect different plan metric variables. The *painting* action increases the value of *labour*. In contrast, the *soak-chemical* action raises the value of the *hazard* variable. In addition to the plan metric, both actions may cause a different value of plan length. The plan length for achieving the similar numeric goal, *(≥(metal-part)1)*, is perhaps longer in the plan that chooses the *soak-chemical* action compared to the plan that incorporates the *painting* action. Figure 3.7 shows there is a preceding action before *soak-chemical* can be selected.

Resource choices also occur at the final step, or in the product assembly process. As depicted in Figure 3.7, *product1* can be constructed using three different actions. Meanwhile, the assembly process of the *product2* is performed by two different actions. These action differ not only in terms of the resources but also the mechanism used to perform the assembly. Some action choices conduct the assembly manually which causes an increase in the *labour* plan metric. Whilst some other actions perform the assembly task using a machine which results in an increase to the *machine* cost.

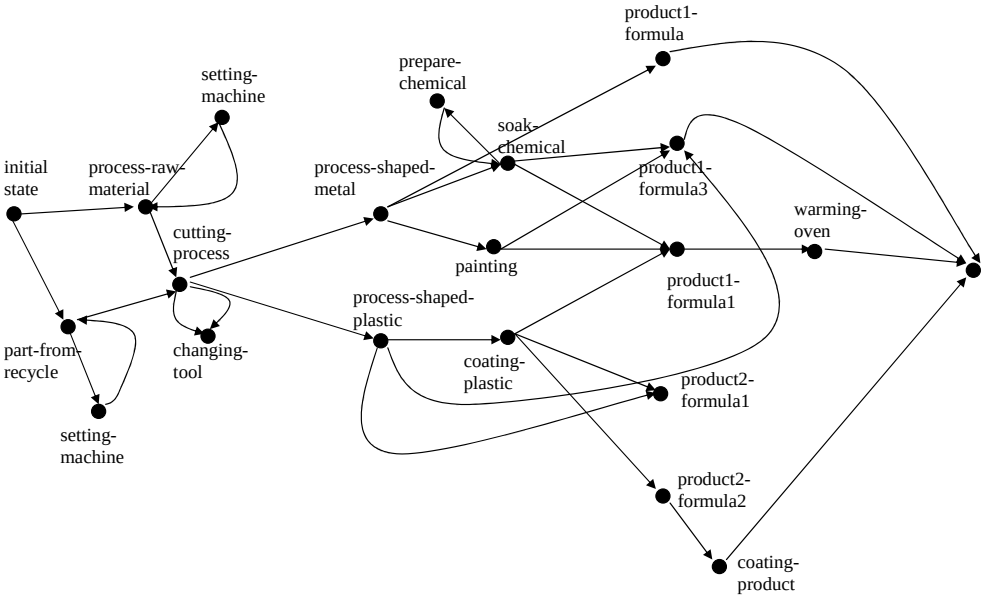

Figure 3.7: General Action Sequences in Production Domain

Table 3.5 and Table 3.6 give the summary of some actions particularly those which incorporate the choice of resources. These actions generally achieve similar numeric states although with different values. They also modify the values of different plan metric variables. The complete domain encoding is given in Appendix D.

3.3.5 Sugar-Supply Domain

The *Sugar-Supply* domain basically combines the manufacturing and logistic problems. In this domain, a product called sugar, is produced in a variety type called *brand* through an action called *produce-sugar* to fulfill the demands from several Depots. The *produce sugar* action is accomplished using several facilities, called *mill*. Each *mill* can process several different types of *brand* at different *mill-cost* and capacity. However, every mill can only process a single *brand* at one time. A set-up process has to be performed in order to change the *brand* type from one to another. Furthermore, each facility is constrained by the different number of set-up activities. The plan metric variables defined in the domain include *mill-cost*, *inventory-cost* and *handling-cost*

Every *brand* type require a similar raw material called *sugar-cane*. The supply of the *sugar-cane*, if needed, can be obtained from the farm. It also can be requested from the other facilities, provided that the facilities have an excessive stock of the *sugar-cane*.

Table 3.5: Production Domain plan metric profiles

action	metric		
	hazard	labour	machine-cost
process-raw-material	1		
part-from-raw material			
painting		1	
soak-chemical	1		
product1-formula1			10
product1-formula2		1	
product1-formula3	1		
product2-formula1			10
product2-formula2		1	

These resource choices are encoded in the actions called *sugar-cane-farm* and *sugar-cane-mills*. The *sugar-cane* is considered a perishable item, so that it is preferable to finish the excessive stock in other facilities before obtaining it from the supplier. To model this situation, both *sugar-cane-farm* and *sugar-cane-mills* actions increase the plan metric variable called *inventory-cost*. However, the *sugar-cane-farm* action increases the variable by a bigger value compared to the *sugar-cane-mills* action. In addition, both actions also increase the amount of the *sugar-cane* by different quantities.

The sugar products have to be loaded into the truck before they can be delivered to the customers at the Depots. Choice of resources is also encoded at this point whereby the loading activities can be performed either manually or using a crane. The encoded actions related to these choices are called *load-truck-crane* and *load-truck-manual*. Both actions load a different quantity of the sugar products and affect the similar plan metric variable called *handling-cost* but with the different quantity. Figure 3.8 illustrates the possibles flow of action sequences in achieving goals.

The encoded domain has the potential to generate alternative solutions including solution giving the optimal plan metric value. For example, if the plan metric encoded is to minimise the *mill cost*, the solution developed must search for the mills that give the minimum cost for a given brand at a given quantity. Sometimes, it is also more cost saving to produce more than one brand using a single facility as the demand can be fulfilled by a single facility. Table 3.7 summarises the alternative actions encoded in the domain that reach a similar numeric state variable as well as given different value to the plan metric variables. The complete domain encoding is given in Appendix E.

Table 3.6: Numeric preconditions and effects in Production Domain

action	numeric precondition						numeric effect					
	rcp	rcm	mp	pp	cp	cm	p1	p2	mp	pp	mt	pl
process-raw-material											4	6
part-from-recycle	2	3									1	1
painting					1				2			
soak-chemical						2		2				
product1-formula1			2	2			2					
product1-formula2			1		1		1					
product1-formula3		1			1		1					
product2-formula1				2	1			3				
product2-formula2	2			4				4				

notes :

rcp	$recycle - plastic$
rcm	$recycle - metal$
mp	$metal - part$
pp	$plastic - part$
cp	$clean - plastic$
cm	$clean - metal$
$p1$	$product1$
$p2$	$product2$
mt	$metal$
pl	$plastic$

3.4 Conclusion

The class of domain-specific metric planning problems is mainly characterised by the choice of resources. The choice of resources is then manipulated by actions in the domain so that they effect the different value of the plan metric variables encoded in the problem. The resource choices will lead to the construction of alternative solutions according to the value of the plan metric. The plan metrics defined in the problem instances are used to guide the search so that good quality solutions are selected. This means the search for the plan focuses on the plan quality aspect. The extended planner in this thesis is designed to solve this class of problem by including the plan metric in its heuristic to obtain the actions with the cheapest cost relative to the value of the plan metric. Apart from having different values of plan cost, the solutions for domain-specific metric problems could be differ in terms of the number of actions or the value of the plan length. The plan length consideration is also included in the heuristic developed for the extended planner.

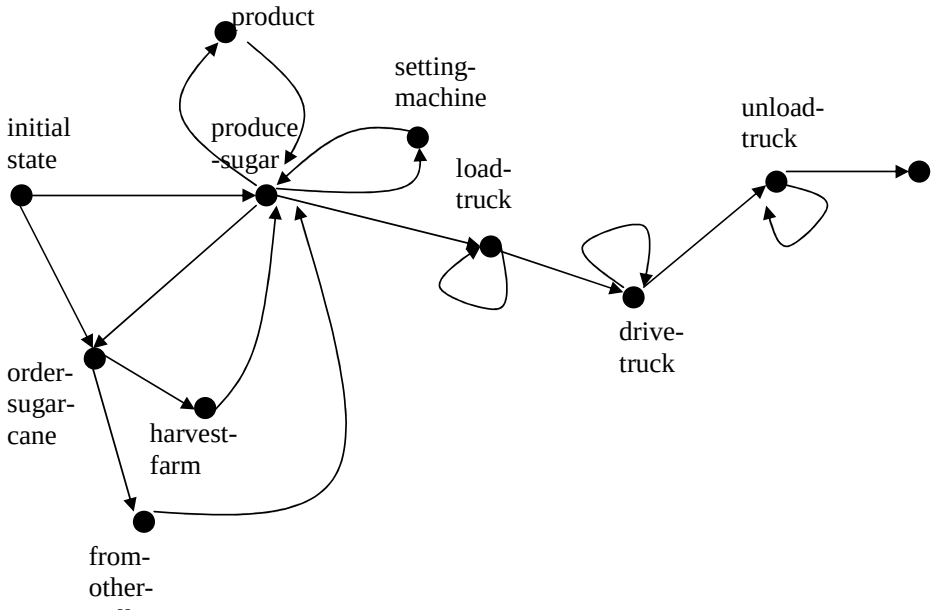

Figure 3.8: General Action Sequences in Sugar Domain

The domains constructed in this chapter include all the above characteristics. They are similar in terms of all domains included with resource choices and featuring the producer and consumer actions. However, each domain presents a different complexity of resource choices at a particular state. The complexity of resources does not only reflect on the number of resources available at a particular state in the domain, but also the consequence of choosing one of the alternative resources. For example in the *Bread* domain, the numeric variable *dough* increases as a result of the effect of two actions namely *kneed-hand* and *kneed-machine*. Either *kneed-hand* or *kneed-machine* is selected at the applicable state in the domain, it does not lead to other complex choices of action. Both of these actions provide the precondition that makes either *making-bread* or *making bun* applicable. However, in *Production* domain the resource choices at any state can lead to other complex choices of either resource or action. In other words, the production domain is encoded with more producer and consumer actions compared to the *Bread* domain. In general, the level of producer and consumer actions is encoded differently in these domains.

Table 3.7: Numeric Preconditions and Effects in Sugar Domain

action	Plan Metric			numeric effect	
	handling-cost	mill-cost	inventory-cost	sugar-cane	sugar
produce-sugar		given by the mill			
sugar-cane-farm			10	5	
sugar-cane-mills			1	1	
load-truck-crane	10				5
load-truck-manually	1				1

Chapter 4

Background To LPRPG

4.1 Introduction

The Hybrid **L**inear **P**rogramming and **R**elaxed **P**lanning **G**raph or **LPRPG** [10] is a numeric domain-independent planner that is being extended in this thesis. This planner is developed on the basis of Metric-FF. It employs the planning graph based heuristic [4] but improves the handling of the numeric part of the problem through the implementation of Linear Programming (LP). Linear Programming is generally known as a powerful optimisation tool and has been used for a long time particularly in the Operational Research (OR) community. It is used to solve optimisation problems with a linear objective function and real valued variables. In Mixed Integer Programs (MIPs) some of the known variables can be restricted to integer values. In the planning community, many researchers exploit LP approaches, [3, 42, 55, 60, 63]. However these works ignore many interesting aspects of the numeric behaviour of the domain.

In LPRPG, the LP is constructed alongside the construction of the Relaxed Planning Graph (RPG) for each evaluated state, S, in the domain. Since it only concentrates on the numeric part, the variables in the LP model consist only of the numeric variables. These are the numeric preconditions and the effects of action variables in each action layer. The numeric variables are real-valued whilst the action variable can take only the values 0 or 1 to indicate whether the action is applicable or not with regard to the current numeric values at the fact layer. The fact that the RPG contains both real and integer variables might suggest that a MIP is most appropriate. However, LP is used since LPRPG relaxes the solution of extraction problem into one in which only real values are present. A detailed explanation of LP construction in a domain is explained in section 4.2.

The implementation of the LP in LPRPG overcomes the problems resulting from ignoring negative delete effects. This is the approach taken by Metric-FF, and it results in poor estimates of the numeric values of the state. This leads to the problems identified by Coles et al; called Helpful Action Distortion (HAD) and Cyclical Resource Transfer (CRT) [10]. These problems often occur in resource intensive planning domains particularly whenever the *producer* and *consumer* actions are comprehensively encoded. Definition of these actions are given in [10]. The following section 4.1 discusses the HAD and CRT issues encountered in numeric domains using Metric-FF.

4.2 Metric-FF

Metric-FF [38] is the successor of the Fast Forward (FF) planner [40] that handles numeric planning problems. It extends the relaxation introduced in [5, 48], to handle numeric variables. The numeric negative effects are ignored during the construction of the relaxed planning graph. Ignoring the delete effects in propositional planning simplifies the solution development and results in a relaxed reachability estimate. However, deleting negative numeric effects can throw away too much information and result in poor search control. According to [38], the numeric planning task can be defined as the following:-

Definition 8 (Numeric Planning Problem) *A numeric task is a tuple* $\langle V, P, A, I, G\rangle$ *where* V *and* P *are the numeric variables and propositions used,* A *is a set of actions that can have both propositional and numeric effects,* I *is a state expressed in terms of both facts and numeric quantities, and* G *is a condition on both propositions and numeric quantities. A sequence of actions* $\langle a_1, \ldots, a_n\rangle \in A^*$ *is a plan if the result of applying it to* I *yields a state that models* G*, ie: result (* I *,* $\langle a_1, \ldots, a_n\rangle)) \models G$*.*

Definition 9 (Numeric Planning Problem) *The numeric task is a tuple* $\langle V, P, A, I, G\rangle$ *where* V *and* P *are the variables and propositions,* A *is a set of actions,* I *is a state and* G *is a condition. A state,* $s = (p(s), v(s))$ *where* $p(s) \subset P$ *and* $v(s) = (v^1(s), \ldots, v^n(s))$ *for all pair of states,* s *and* $s^{'}$*, effect of* v^i*,* $\{increase, decrease =\}, p.q$ *where* p *is the vector of numeric variables and* q *a vector of constants.*

In comparison to STRIPS, a set V of numeric variables is added to numeric task in which $V = v^1, \ldots, v^n$. A state, s, in a numeric task is a pair $s = (p(s), v(s))$ where $p(s) \subset P$ and $v(s) = (v^1(s), \ldots, v^n(s)) \in \mathbb{Q}^n$ is a vector of rational numbers. A relaxed planning graph is built starting from state s, in a numeric task $\langle V, P, A, s, G\rangle$. Metric-FF [38] expresses numeric problems in *linear normal form* (LNF) where, $\forall v^i : v^i(s) \leq v^i(s^{'})$.

In achieving this, only positive variables are considered. Therefore, an inverted variable, $-v^i$, or $(-1) * v^i$, is used to replace each variable v^i where v^i is decreasing. In building the relaxed planning graph, each variable v^i is in the value range [lower bound, upper bound]. The upper bound of numeric variables in the fact layer are updated using the total sum of the increasing effects of the layer. The upper bound of $-v^i$, corresponds to the negation of a lower bound on v^i . This approach to estimating values is referred to as *bounds propagation*.

Definition 10 (Bounds Propagation Technique) *Given variables $v_0, v_1 \ldots, v_n$ at each layer $l_0, l_1 \ldots l_n$: the upper and lower bounds of the numeric variable, v_i, are updated using bound propagation in which :*

upper bound $= v_{i,l} = v_{i,l-1} + \sum_{x=0}^{n} \mathrm{pa}_{x,l}$;
lower bound $= v_{i,l} = v_{i,l-1} + \sum_{x=0}^{n} \mathrm{ca}_{x,l}$.

The *pa* are all actions with a positive numeric effect on $v_{i,l}$ and *ca* are all actions with a negative numeric effect on the variable $v_{i,l}$.

To explain bounds propagation, suppose a domain called *supply-chain* has four actions, namely *produce*, *load*, *unload* and *drive* as illustrated in Figure 4.1. In brief, the domain models a simple production activity together with a logistic system whereby an item is produced by a facility at a location, *store*, and then transported to another location, *site*, by a truck. Apart from those actions, there is a set of numeric variables = $\{ft, st, tk\}$ defined in the problem. The ft, st and tk variables represent the quantity of the item at the store, site and truck respectively. The role of the action *produce* is to produce the item and increase the value of the variable ft. The action *load* decreases the value of ft and increases the value of tk. Another action, *unload*, increases and decreases the value of st and tk respectively. Importantly, this domain structure presents a significant numeric interaction between *consumer* and *producer* entities.

In this *supply-chain* domain we assume the variables will increase or decrease by the application of a particular action at the rate of 1 unit at a time. The initial value of $ft = 2$ and the numeric goal for this problem is set as $st \geq 2$. The bounds calculation is as follows. The upper bound for variable ft_1 is calculated by, $ft_1 = ft_0 +$ *produce*(p_1)=2+1=3. Whilst for the lower bound, $ft_1 = ft_0$ - *load*(l_1)=2-1=1. To solve this problem, MetricFF will develop the relaxed planning graph, layer by layer, until it reaches the goal state. The layer construction and numeric estimation at every layer, using bounds propagation, is shown in Figure 4.2. The planning graph will expand until layer 3 since the goal is considered achievable at this layer.

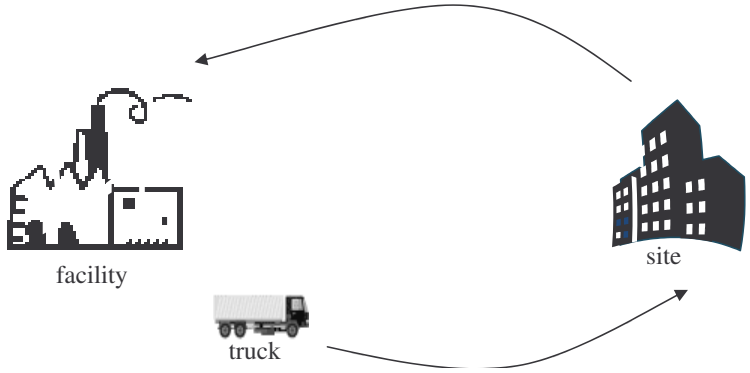

Figure 4.1: Simple Supply Chain Domain

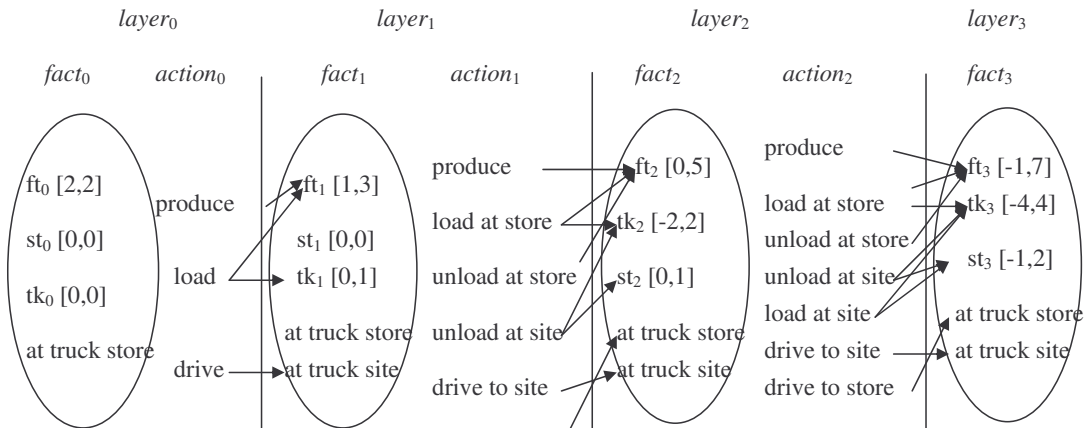

Figure 4.2: Bounds Propagation in the Relaxed Planning Graph for Supply Chain Domain

This technique results in an inaccurate computation on the numeric values. The upper and lower bounds for each numeric diverge, as shown in Figure 4.2. For example, the value of ft_3 is in the range of $[-1, 7]$. Moreover, the negative value does not provide any meaningful information about the resources available. Realistically, a resource can hold at minimum a zero value which represents unavailability. In order to obtain a better estimate of the value, suppose the negative numeric effect is considered in estimating the bounds value. Using the same initial values and planning graph, another bounds calculation is conducted manually from the initial layer until the goal layer. The new estimated bounds are presented at column 3 in Table 4.1. The bounds of the numeric variables are now more realistic compared to those obtained from the bounds propagation approach. This also suggests, in general, that the negative numeric effects should be taken into account to obtain more accurate estimates of the numeric bounds.

Table 4.1: Difference Between Bounds Propagation and Manual Calculation

variables	bound propagation	manual calculation
ft_0	$[2, 2]$	$[0, 2]$
st_0	$[0, 0]$	$[0, 0]$
tk_0	$[0, 0]$	$[0, 0]$
ft_1	$[1, 3]$	$[0, 2]$
st_1	$[0, 0]$	$[0, 0]$
tk_1	$[0, 1]$	$[0, 1]$
ft_2	$[0, 5]$	$[0, 3]$
tk_2	$[-2, 2]$	$[0, 1]$
st_2	$[0, 1]$	$[0, 1]$
ft_3	$[-1, 7]$	$[0, 4]$
tk_3	$[-4, 4]$	$[0, 1]$
st_3	$[-1, 2]$	$[0, 2]$

In Metric-FF[38], the relaxed planning graph is extracted in the backward approach which means it started from the goal layer. An achiever for facts at the goal layer, $layer_n$, creates preconditions that have to be satisfied at $layer_n - 1$. This backwards plan extraction is repeated until the first layer is reached. The heuristic value is the total number of actions required to reach the goal state from the initial state. However, actions appearing in the first layer of the relaxed planning graph are considered to be *helpful actions*, which are preferred for constructing the successor states in the search. Poor estimates of numeric bounds can influence the heuristic calculation. An action is considered as an achiever for a particular fact if all of its propositional and numeric preconditions hold [10]. The numeric precondition of actions at layer l are satisfied by the upper bounds of the corresponding numeric variables at layer *l-1*. Therefore in the above example, the numeric precondition can always be satisfied since the negative delete effect only reduces the lower bound of the variable. The consequence of this situation is that a resource can always be consumed without it being necessary to execute the action that produces or increases the resource. For example, the *produce* action in Figure 4.1 produces 1 unit of resource, which causes variable ft to increase in value. This variable is consumed by the action *load* provided that the precondition $ft > 0$ is satisfied. Since the execution of the *load* action does not result in any changes to the upper bound of variable ft, this indirectly indicates the action *load* can be executed as many times as required without it being necessary to place the *produce* action in the plan. The specific term used to describe this situation is called Resource Persistence (ReP) [10]. *ReP* also happens in propositional domains but only results in an underestimate of the work required to achieve a fact. It can cause problems in

numeric domains if the numeric interaction between so called *producer* and *consumer* actions is intensively encoded in a domain. In brief, in the above supply chain domain, the *produce* action is classified as a producer action whilst the *load* action is classified as a consumer action.

The ReP situation causes the relaxed plan to contain no producer action since its precondition cannot be achieved without the consumption of the resources. This leads to a problem called *Helpful Action Distortion* or (HAD). The HAD means the *producer* action will not include producer actions as helpful actions even though the current state is one in which producers can conveniently be applied. This can result in dead-ends. Another phenomenon created by the Metric-FF heuristic is called Cyclical Resource Transfer (CRT) [10]. For example, the valid relaxed plan might include the following:-

```
0.01: (load item truck store)
1.02: (unload item truck store)
```

In the above relaxed plan, the first action increases the resource, `item`, in the *truck* by 1 and the second action exploits the item to be unloaded from the `truck` back to the `store`. As a result the number of `items` at the `store` is two, because the action *unload* behaves like a producer action. CRT is a phenomenon where a resource is produced by an artificial artifact of the relaxation [10]. In sum, ignoring negative delete effects can lead to no solution being found, particularly in numeric domain featuring complex numeric interactions.

4.3 LRPPG

LPRPG has a common structure with Metric-FF whereby both planners apply the relaxed planning graph and ignore the negative delete effects of the propositional part during the construction of the solution. But in contrast to Metric-FF, LPRPG includes the negative numeric effects when estimating the numeric bounds in every layer. In order to do the estimation, LPRPG exploits Linear Programming (LP), a powerful optimisation tool, to obtain both upper and lower bounds of numeric variables in every layer. The LP is further used in the relaxed plan extraction phase to acquire the minimum number of actions in a particular relaxed plan. The LP solver is called to obtain the solution of every LP model that is being constructed during the solution development.

4.3.1 Relaxed Planning Graph Construction

The relaxed planning graph construction alternates between the fact and the action layers. The LP models in LPRPG [10] are constructed along with these layers in every evaluated state. The LP model includes the numeric precondition variables in the fact layer and all action variables in the action layer. The numeric variables take real values whilst the action variables accept integer values or precisely the binary 0 or 1 values. The 0 value indicates that the action is not applied whereas the 1 value implies the action is applied. However to relax the solution, the action variables will take values within the range $[0, 1]$ to indicate a partial application of an action in the relaxed plan. Similar to Metric-FF, every numeric variable has upper and lower bounds. The lower bound results from the minimisation of the objective function whereas the upper bound will correspond to the maximisation of the objective function for a numeric variable. Therefore, the LP solver is called twice in doing the bounds calculation. The constraints are constructed based on the applicability of actions to a particular variable at a particular layer. The actions are categorised as *producer* or *consumer* actions. These actions are identified prior to the construction of the planning graph by an inference module called TIM [23]. The producer *action* basically increases the numeric value. Meanwhile, the *consumer* action decreases the numeric value. By modeling the effects of these actions in the LP, the true flow of resources is modelled and not just their accumulation as triggered in the bounds propagation technique.

Figure 4.3 demonstrates the construction of the LP model at the first layer for the above supply chain problem. As can be seen, there are three numeric variables; item at the facility, ft_0, item in the truck, tk_0 and item at the site, st_0; and three action variables; action *produce*, pr_1, action *load*, ld_1, and action *drive* dr_1.

However, only two actions are considered applicable which means their preconditions are satisfied at this layer; pr_1 and ld_1. Application of these actions give increasing and decreasing effects to variables ft_1 and tk_1 at layer 1. Therefore, LP models to determine the upper and lower bounds of variables ft_1 and tk_1 can be constructed as shown in Figure 4.4 and Figure 4.5. The constraints developed for ft_1 can be read as, the value ft_1 at layer 1 is equal to its initial value, which is 2, plus the number of item produced layer 1, pr_1, minus the number of item that being loaded into the truck, ld_1. In the simplest form the value of $ft_1 = ft_0 + pr_1 - ld_1 = 2 + 1 - 1 = 2$. Therefore, as can be seen from Figure 4.3 the value of ft_1 is [0,2]. But, the LP model has to be written in the constraint form as presented in Figure 4.4. The value of variable st_1 at layer 1 is similar to st_0, value at layer 0, since no action is applicable to this numeric variable at this layer. The new estimates of the upper and lower bounds at layer 1 help to select

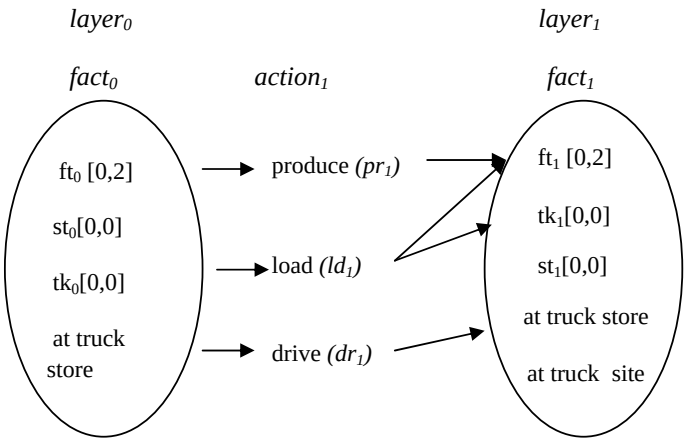

Figure 4.3: LP Estimation for the First Layer in the Relaxed Planning Graph

$$
\begin{aligned}
&\text{Min/Max } ft_1 \\
&\text{subject to :} \\
&ft_1 - 2 - pr_1 + ld_1 = 0 \\
&0 \leq pr_1 \leq 1 \\
&0 \leq ld_1 \leq 1 \\
&ft_0, ft_1 \geq 0
\end{aligned}
$$

Figure 4.4: LP Model for numeric variable ft_1 at layer 1

the next applicable actions of the following layer. The construction of LP models will continue until it reaches the goal state.

The overall construction of the relaxed planning graph layers and their estimated numeric bound values using LP is illustrated in Figure 4.6. As shown, the upper and lower bounds of each numeric variable are more realistic and informative compared to the value obtained from the bounds propagation technique. See Table 4.1 for comparison. The new bounds estimate from the LP are found similar to those manual

$$
\begin{aligned}
&\text{Min/Max } tk_1 \\
&\text{subject to :} \\
&tk_1 - tk_0 - ld_1 = 0 \\
&0 \leq ld_1 \leq 1 \\
&tk_0, tk_1 \geq 0
\end{aligned}
$$

Figure 4.5: LP Model for Numeric Variable tk_1 at Layer 1

calculated values.

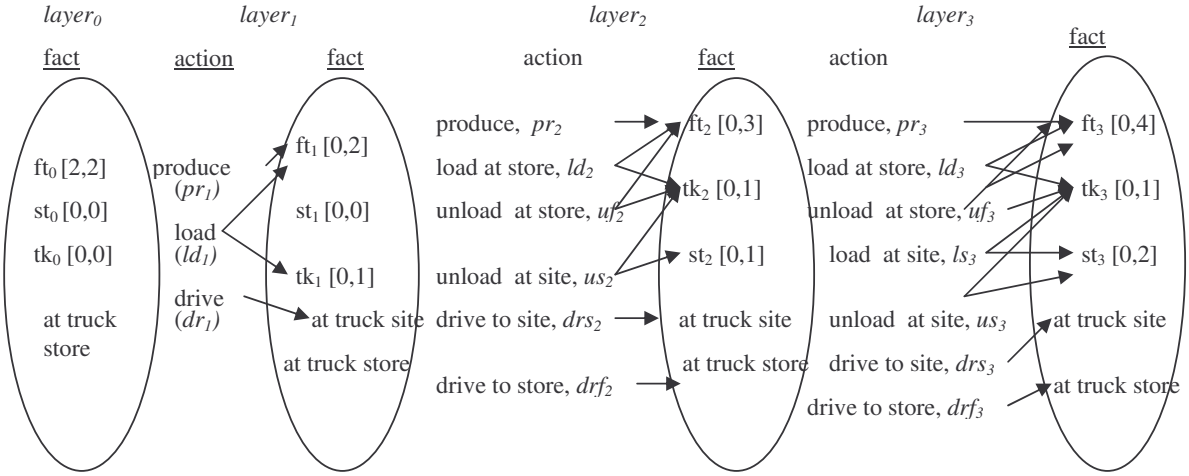

Figure 4.6: LP Estimation for all Numeric Variables in all Layers of the Relaxed Planning Graph in Supply Chain Domain

The size of the LP model increases as the layer increases since the number of applicable actions increase. For example, the LP model constructed for ft_3 at layer 3 is shown in Figure 4.7. The overall constraints that constructed during the planning graph expansion are depicted in Figure 4.8

4.3.2 Relaxed Planning Graph Extraction

Unlike Metric-FF, LPRPG does not apply backward plan extraction to obtain the heuristic value, but uses LP instead. The heuristic value is still computed as the number of actions in the relaxed plan. To acquire this, a queue of goal layers is defined ordered from the deepest first or the goal at the final layer. During regression, for each variable in a precondition, a linear programming solver is used to find the actions that contribute to the variable meeting its bound at the appropriate layer in the graph. All actions are weighted, and these weights are used in determining the heuristic value.

$$
\begin{aligned}
&\text{Min/Max } ft_3 \text{ subject to :}\\
&ft_3 - ft_2 - pr_3 + ld_3 - uf_3 = 0\\
&ft_2 - ft_1 - pr_2 + ld_2 - uf_2 = 0\\
&ft_1 - ft_0 - pr_1 + ld_1 = 0\\
&ft_0 = 2,\ ft_1, ft_2 \geq 0
\end{aligned}
$$

Figure 4.7: LP Model for Numeric Variable ft_3 at layer 3 in Supply Chain Domain

$$
\begin{aligned}
&ft_2 - ft_1 - pr_2 + ld_2 - uf_2 = 0\\
&tk_2 - tk_1 - ld_2 + us_2 + uf_2 = 0\\
&st_2 - st_1 - us_2 = 0\\
&ft_3 - ft_2 - pr_3 - uf_3 + ld_3 = 0\\
&tk_3 - tk_2 - ld_3 + uf_3 - ls_3 + us_3 = 0\\
&st_3 - st_2 - us_3 + ls_3 = 0
\end{aligned}
$$

Figure 4.8: Numeric Constraints at Layer 2 and 3 in Supply Chain Domain

The weight given to an action is 1.1^n , where n is the layer at which the action first appears in relaxed planning graph. LPRPG also adopts an approach similar to that of Metric-FF in preferring the helpful actions. The way weights are assigned to each action variable in the LP objective function results in the helpful actions being chosen from the actions that appear in the first layer. The LP objective function will minimise the weighted sum over the action variables. The constraints will include the overall constraints constructed from first to goal layer. Therefore, the LP model for relaxed plan extraction for the supply-chain example can be seen in Figure 4.9. The LP solver will return the number of actions in the relaxed plan as well as the value of the action variable. The action with non-zero value will be considered to be included among the actions in the relaxed plan.

Although the use of the LP in LPRPG greatly improves its heuristic estimates, and leads to better solutions than are constructed by Metric-FF in domains where the producer and consumer actions interact, LPRPG still has an important weakness. Just as in Metric-FF, the solution constructed by LPRPG aims to minimise the number of actions. It excludes the plan metric in its solution development. This thesis extends LPRPG to address optimisation of the domain-specific metric.

Min $1.1^1 pr_1 + 1.1^1 ld_1 + 1.1^2 pr_2 + 1.1^2 ld_2 + 1.1^2 uf_2 + 1.1^2 us_2 +$
$1.1^3 pr_3 + 1.1^3 ld_3 + 1.1^3 uf_3 + 1.1^3 ls_3 + 1.1^3 us_3$

subject to :

$ft_3 - ft_2 - pr_3 - uf_3 + ld_3 = 0$
$ft_2 - ft_1 - pr_2 - uf_2 + ld_2 = 0$
$ft_1 - ft_0 - pr_1 + ld_1 = 0$
$tk_3 - tk_2 - ld_3 + uf_3 - ls_3 + us_3 = 0$
$tk_2 - tk_1 - ld_2 + us_2 + uf_2 = 0$
$st_3 - st_2 - ls_3 + us_3 = 0$
$st_2 - st_1 - us_2 = 0$
$tk_1 \geq 0,\ tk_2 \geq 0,\ tk_3 \geq 0$
$ft_1 \geq 0,\ ft_2 \geq 0,\ ft_3 \geq 0$
$st_2 \geq 0,\ st_1 \geq 0$
$0 \leq pr_1, pr_2, pr_3 \leq 1$
$0 \leq ld_1, ld_2, ld_3 \leq 1$
$0 \leq ls_3 \leq 1$
$0 \leq uf_2 \leq 1,\ 0 \leq uf_3 \leq 1$
$0 \leq us_2 \leq 1,\ 0 \leq us_3 \leq 1$

Figure 4.9: LP Model for Relaxed Plan Extraction in Supply Chain Domain

4.4 Conclusion

LPRPG [10] is the basis of the extended planner developed in this thesis. It shares a common structure with Metric-FF but solves the problems that result from ignoring negative numeric delete effects. It uses Linear Programming to do the bounds calculation for the numeric variables. This approach gives a better estimate of the bounds value compared to the value given by the bounds propagation technique that is applied in Metric-FF. Improved estimates of the bounds helps to solve the Helpful Action Distortion (HAD) and Cyclical Resource Transfer (CRT) problems since it gives more accurate information about the resource availability. LPRPG applies Linear programming in the plan extraction phase in which the solution minimises the number of actions in the relaxed plan. However, LPRPG does not take domain specific metric functions into account so, even when these are supplied as part of the domain specification, LPRPG is unable to do any better than to attempt to minimise the length (and not the cost) of the plan. The key novelty in this thesis is the extension of LPRPG to handle domain specific metric functions and thereby balance plan length against plan cost.

Chapter 5

MetricLPRPG

5.1 Introduction

MetricLPRPG is a planner developed in this thesis that extends the LPRPG planner to include plan quality in solution construction. Plan quality is an important issue that is captured in the domain-specific metric class. The domain-specific metric class is a class of metric planning problems enriched with expressive metric functions. The metric function is referred to as the *plan metric*, an important extension made in PDDL2.1 [24]. The *plan metric* represents how the *plan cost* should be determined. Therefore, parameters in the *plan metric* can include resources that are being used and considered in the planning problem, such as labour, energy, money, etc. These parameters are essential and often included in many realistic planning problems. The *plan metric* extension enhances the capability of modeling application problems, however, it increases the complexity of the solution development. In the same initial and goal states problem, the availability of resource choices results in various solutions relative to the objective function. Importantly, the plan evaluation for such problems incorporates the *plan cost*, rather than solely depending on the plan length.

A simple example to demonstrate the behavior of action and resource choices that affect the plan length and cost is a washing laundry problem. The choices available for this task are either doing it by hand or using a machine. The first choice increases the labour cost, whilst the second choice may increase the energy cost. The energy cost is often higher than the labour cost. However, washing by machines might result in a shorter plan. This is due to the fact that the washing machine can wash in a bigger quantity compared to manual washing. For this reason, for a certain quantity of the laundry, there might be fewer *wash* actions in the plan that chooses the machine than in the plan that chooses washing manually. In addition, washing by machine usually

takes out some steps that are performed if washing manually. This also contributes to shortening the plan length. Following this, the plans with longer plan length perhaps have lower cost, shorter plans have higher cost. Chapter 3 discusses and gives examples of this problem. In general, many cases of planning with resources often have a trade off between the plan length and the plan quality. The trade-off further increases the solution complexity. The ideal solution is an optimal solution that balances the plan length and plan quality values.

MetricLPRPG, basically implements the idea of balancing between the value of the plan length and plan metric. The solutions developed have a reasonable plan length with the best achievable plan cost. This is achieved by extending the objective function of the Linear Programming used in the plan extraction phase, so that it includes the metric variables used in the plan metric function. By adding the plan metric, the new Linear Programming objective function now attempts to minimise both plan length and plan cost. This corresponds to Linear Programming with Multi-objective functions (MOLP). The weighted-sum technique which is discussed in detailed in Chapter 2 is adopted in order to obtain the Pareto-optimal solution of the MOLP. However, only a single, Pareto optimal is obtained within the implementation of this technique. Beside extending the Linear Programming objective function, the plan metric variables are included during the relaxed plan construction. The bounds of the plan metric variables are estimated as well as those of the numeric preconditions. The estimated values of the plan metric further helps to choose actions with minimum cost resources. This chapter explains the implementation of the plan metric variables during the relaxed plan construction and extraction in MetricLPRPG.

5.2 Construction of Multiple Solutions in Metric Domain

Domain specific metric problems as mentioned before, have the potential to generate more than one solution that attempts to optimise the objective function value. To demonstrate this, some modifications have been made in the Settlers Domain in IPC3 [46] as explained in Chapter 3. This domain has interesting numeric interactions in which it exhibits the behavior of *producer* and *consumer* actions but lacks action choices that effect the plan metric. It is important to add alternatives choices of action and resource in the domain. The modified version is called the Extended Settler Domain. The Extended Settler Domain has two new actions: *build-wood-house* and *fell-timber-machine*. Both actions achieve the same goal but have different numeric effects and therefore different effects in the plan metric. The new actions are encoded as illustrated in Figure 5.1.

```
 (:action build-wood-house
  :parameters (?p - place)
  :precondition (and (>= (available wood ?p) 2))
  :effect (and (increase (housing ?p) 1)
(decrease (available wood ?p) 2))
 )

  (:action fell-timber-machine
   :parameters (?p - place)
   :precondition (has-cabin ?p)
   :effect (and (increase (available timber ?p )3)
        (increase (pollution)3))
  )
```

Figure 5.1: Alternative Actions in Extended Settler Domain

The *build-wood-house* is an alternative action to the existing *build-house* action. Whereas, the *fell-timber-machine* action is an alternative to the existing *fell-timber* action. The existing *fell-timber*, *build-house* and *break-stone* actions are encoded in the Settler Domain as depicted in Figure 5.2. The complete code for the Extended Settlers Domain is given in Appendix A. The plan metric variables encoded in the problems constructed for the Extended Settler Domain are similar to those that have been encoded in the Settlers Domain. The plan metric variables are *pollution*, *resource_use* and *labour*. The values of these plan metric variables are increased whenever applying certain actions in the domain. For example, as illustrated in Figures 5.1 and 5.2, the execution of the *fell-timber-machine* action increases the *pollution* value. Whereas, the *fell-timber* action increases the value of the *labour* variable. Instead of increasing the value in different plan metric variables, different alternative actions might require different numeric preconditions or resources. For example, the application of the *build-wood-house* action requires only *wood* compared to the existing *build-house* action that demands both *wood* and *stone* as the resources.

In order to describe the possible solutions generated for the Extended Settler Domain, the goal for such problems is set to ($\geq$ *(housing at location1)* 1). All relevant numeric variables required to achieve this goal are initialised to zero value. Suppose, the possible relaxed plans that achieve the stated goal are depicted in Figure 5.3, 5.4 and 5.5. As shown, these relaxed plans include different actions that result in different values of the plan length and plan cost .

```
 (:action build-house
   :parameters (?p - place)
   :precondition (and (>= (available wood ?p) 1)
(>= (available stone ?p) 1))
   :effect (and (increase (housing ?p) 1)
(decrease (available wood ?p) 1)
(decrease (available stone ?p) 1)))

 (:action fell-timber
   :parameters (?p - place)
   :precondition (has-cabin ?p)
   :effect (and (increase (available timber ?p) 1)
(increase (labour) 1))
   )

  (:action break-stone
   :parameters (?p - place)
   :precondition (has-quarry ?p)
   :effect (and (increase (available stone ?p) 1)
(increase (labour) 1)
(increase (resource-use) 1)
))
```

Figure 5.2: Existing Actions in Settlers Domain

```
0.001 : fell-timber location1
1.002 : fell-timber location1
2.003 : build-wood-house location1
```

Figure 5.3: Potential Relaxed Plan for Extended Settler(1)

```
0.001 : fell-timber-machine location1
1.002: build-wood-house location1
```

Figure 5.4: Potential Relaxed Plan for Extended Settler(2)

```
0.001 : fell-timber location1
1.002 : break-stone location1
2.003 : build-house location1
```

Figure 5.5: Potential Relaxed Plan for Extended Settler(3)

Referring to Figure 5.3, 5.4 and 5.5; which of these figures can be selected as the solution?. As mentioned previously in the various sections, the domain modeler is given an opportunity to influence the quality of solution according to their preference through the plan metric. They can experience the consequence of applying different coefficients that imply their preference for particular plan metric variables, or only

include the preference metric variables in the plan metric. For example, the coefficient of the plan metric variables in the problem for the above Extended Settler Domain are given as the following;

```
(:metric minimise (+ (+ (* 3 (pollution)) (* 2 (resource-use)))
(* 0 (labour))))
```

These coefficients indicate that the domain modeler prefers a plan that consists of actions that increase *labour* costs to action that raise the value of either *pollution* or *resource-use* variables. Assume that the possible solutions or generated plans with regard to this plan metric are similar to the relaxed plans constructed in Figure 5.3, 5.4 and 5.5. The manual calculation of the plan metric values for these relaxed plans shows that the plan costs are 0, 9 and 2 for the solutions in Figure 5.3, 5.4 and 5.5 respectively. Based on these values, the solution in Figure 5.3 is selected as the best solution since it has the cheapest plan cost. However, the plan length value cannot be simply neglected in the plan evaluation since it has been conventionally used in the previous studies to determine the best plan. Therefore, it is also included in this evaluation and results in the total sum of the plan length and plan cost. The manual calculation of this total sum are 3, 11 and 5 for solutions 1, 2 and 3 respectively. According to these values, the solution in Figure 5.3 again appears to be the best solution with regard to the above plan metric function. The following section discusses the technique employed in MetricLPRPG to find the best solution whenever both plan cost and length have to be considered.

5.3 Implementation of the Plan Metric in MetricLPRPG Heuristic

MetricLPRPG generally inherits the principles applied in its predecessor LPRPG, in constructing the solution. It employs Linear Programming(LP) in the relaxed planning graph construction but extends the objective function of LP in the relaxed plan extraction. The solution developed by LPRPG emphasises plan length, while, MetricLPRPG aims for plan quality without neglecting the plan length aspect. Therefore, the plan metric variables that capture the plan quality aspect of a domain are taken into consideration in developing the MetricLPRPG heuristic. The attempt to include the plan metric variables in the heuristic results in significant impact on the quality of the solution.

5.3.1 Including the Plan Metric Variables in Relaxed Plan Expansion

In addition to the numeric precondition variables, MetricLPRPG incorporates the plan metric variables during the construction of the relaxed planning graph. The LP model is constructed for every plan metric variable at every layer. The upper and lower bounds of the metric variables are obtained from both minimising and maximising the LP objective function. This approach is similar to the numeric precondition bounds calculation conducted by LPRPG. However, the coefficients of the plan metric variables that are commonly stated in the problem, are ignored when doing the bounds calculation. The constraints developed for each plan metric variable at each layer, depends on applicability of actions at a particular layer. The actions are considered applicable whenever their numeric preconditions are satisfied. Thus, MetricLPRPG considers only the plan metric variables that appear as effects of the *producer* or *consumer* actions. It does not include the plan metric variables that are present as effects in the actions that satisfy only propositional facts. Although the constraints for plan metric variables deal with *producer* and *consumer* actions, the constructed LP does not attempt to model the resource flow. It is instead concerned with modeling the level of resource consumption by different action choices. The estimated bounds of the plan metric variables conducted at this phase are useful at the relaxed planning graph extraction. It helps to choose actions which consume minimum resources.

Consider the Extended Settler Domain discussed in Section 5.2. The plan metric variables consist of *labour*, *pollution* and *resource_use*. In this domain, some of the basic facilities have to be established before other facilities can be constructed. The basic facilities do not build up from a specific quantity of resources, but are rather constructed based on propositional facts. For example, *cabin* and *quarry*. In consequence, the early construction of the relaxed planning graph is mainly dealt with propositional facts. Even though these propositional facts are important, they are not being considered in the construction of the LP. Therefore, in the following explanation, some of the important and relevant propositional facts are set to true. Assume the facts of *has_cabin* and *has_quarry* are already established, then the first layer construction of a relaxed planning graph is exhibited in Figure 5.6.

The notations defined in Table 5.1 will be used to represent action, numeric precondition and plan metric variables in constructing the relevant LP model. The general constraint for each numeric facts in Figure 5.6 is presented in Figure 5.7,

The variable with prime notation symbol represents its value at the next layer. As mentioned before, a constraint at a particular layer is constructed based on the

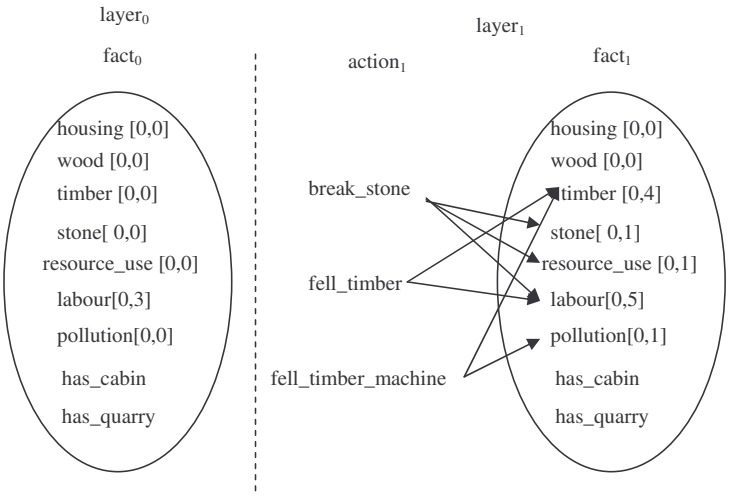

Figure 5.6: Relaxed Planning Graph for Extended Settler at Layer 1

$house' - house - bh - bwh = 0$
$wd' - wd - sw + bh + 2bwh = 0$
$tim' - tim - ft - 3ftm + 2bsm + sw = 0$
$st' - st - bs + bh = 0$
$ruse' - ruse - bs = 0$
$lab' - lab - ft - 2bsm - bs = 0$
$pol' - pol - ftm = 0$

Figure 5.7: General constraint for numeric variables in Extended Settler Domain

actions applicable at that layer. In Figure 5.6, the applicable actions at $layer_1$ involve *break_stone*, *fell_timber* and *fell_timber_machine*. Application of the *break_stone* action, increases value in both *labour* and *resource_use* variables. Meanwhile, the *fell_timber_machine* and *fell_timber* actions increase the value in variables *pollution* and *labour* accordingly. The LP models constructed for these variables are shown in Figure 5.8, 5.9 and 5.10.

Min/Max pol_1
subject to :
$pol_1 - pol_0 - ftm_1 = 0$
$0 \leq ftm_1 \leq 1$
pol_0 , $pol_1 \geq 0$

Figure 5.8: LP Model for Variable *pollution* at Layer 1 for the Extended Settler Domain

Table 5.1: Notations for Variables

variable name	notation
break_stone	bs
fell_timber_machine	ftm
fell_timber	ft
build_sawmill	bsm
saw_wood	sw
build_house	bh
build_wood_house	bwh
pollution	pol
resource_use	ruse
labour	lab
housing	house
timber	tim
wood	wd
stone	st

Min/Max $ruse_1$
subject to :
$ruse_1 - ruse_0 - bs_1 = 0$
$0 \leq bs_1 \leq 1$
$ruse_0$, $ruse_1 \geq 0$

Figure 5.9: LP Model for Variable *resource_use* at layer 1 for the Extended Settler Domain

Min/Max lab_1
subject to :
$lab_1 - lab_0 - ft_1 - bs_1 = 0$
$0 \leq ft_1 \leq 1$
$0 \leq bs_1 \leq 1$
lab_0 , $lab_1 \geq 0$

Figure 5.10: LP Model for Variable *labour* at layer 1 for the Extended Settler Domain

Suppose the numeric goal that has been set in the problem is ($\geq$ *(housing at location1)* 1). The completed relaxed planning graph construction reaches the goal layer as shown in Figure 5.11. As depicted in the Figure, some actions are repeated in the construction of subsequent layers. Conceptually, the LP model is constructed in parallel with the construction of these layers. The repeated actions cause the number of the LP variables to increase. To avoid this, in the implementation, the layer is compressed

so that a single action layer is considered during the expansion [10].

Thus, the layer will contain new actions and actions from previous layers. An action variable is created only if the action is new to that layer. For those actions that are already established from the previous layer, the bound of these layers will be increased to indicate the multiple application of the action. For example, in Figure 5.11, *build_saw_mill* is a new action at layer 2, the range for this variable is [0,1]. Whilst, the *break_stone*, *fell_timber_machine*, *fell_timber* are actions brought from the previous layer. Therefore, the upper bound of these variables range is incremented by 1, thereby their range now are [0,2]. The LP model constructed for variable *labour* at layer 2 is as follows;

Min/Max lab_2
subject to:
$lab_2 - lab' - bs - ft - 2bsm = 0$
$0 \leq ft \leq 2$
$0 \leq bs \leq 2$
$0 \leq bsm \leq 1$

5.3.2 Multi-objective Linear Programming in Relaxed Plan Extraction

MetricLPRPG employs Multi-objective Linear Programming (MOLP) in the relaxed plan extraction. It uses MOLP to work out on how to satisfy all the numeric preconditions with minimum value of plan length and plan cost. The general framework of MOLP involves more than one objective function as explained in Chapter 2. The objective functions often have trade-offs between each other. The conflicting objective functions also happen in the solutions developed for the domain-specific metric problems for which MetricLPRPG is designed. In general, the MOLP model in MetricLPRPG can be stated as follows,

Minimise $\sum f_1(x) + \sum f_2(y)$
where:
$f_1(x); x \in X$; X= action variables
$f_2(y); y \in Y$; Y= plan metric variables

The first function, $f_1(x)$, consists of actions variables required to achieve numeric preconditions. Meanwhile, the second function, $f_2(y)$, represents the resource consumption along the construction of the relaxed plan. The coefficients of the plan metric variables encoded in the problem are taken into account in this model. The ideal solution

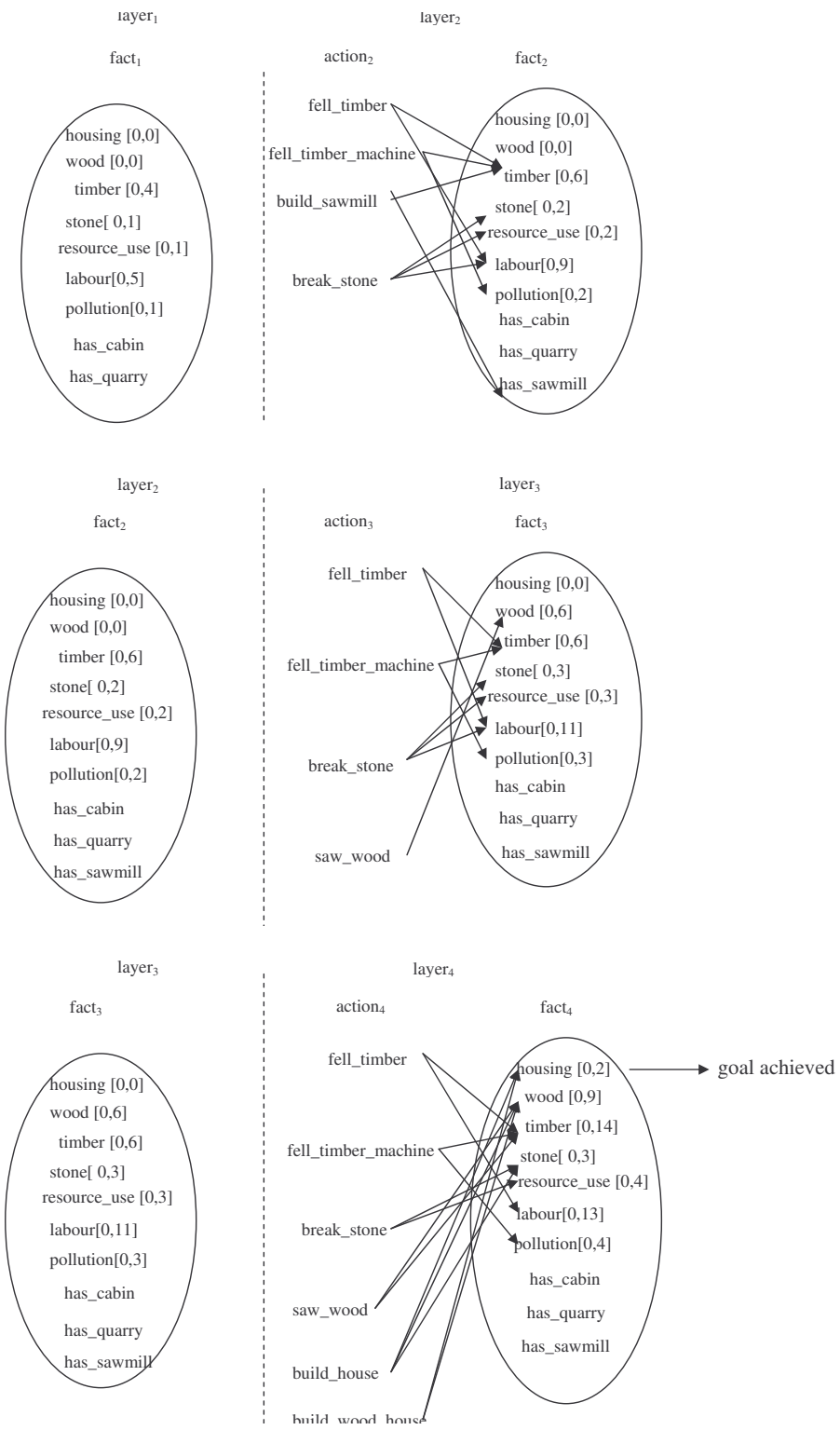

Figure 5.11: Complete Relaxed Planning Graph for Extended Settler Domain

is obtained from the solution of *minimise* $f_1(x)$+ *minimise* $f_2(y)$. Since these functions are traded-off against each other, whilst minimising f_1 it is possible to increase the value of function f_2. The solution that gives the optimal value to each function is difficult to find. However, we could construct an efficient solution in which it can be considered good in terms of every objective and better on one of the objectives. This can be achieved through minimisation of the total sum of both functions. This indicates, both functions are transformed into a single function, which is also known as the *scalarization* approach. The *weighted sum* technique is used to obtain an efficient solution, known as the Pareto optimal. This technique is considered since it is simple but efficient. Furthermore in this technique, a weight is given to every variable to indicate its relative plan metric variables. The weight in the action variables reflects the principle of choosing helpful actions applied in the relaxed planning graph. The weight for each action variable is 1.1^n where n is the layer at which the action is first appeared in the relaxed planning graph. The weights for the plan metric variables are extracted from the problem file. These weights are given by the domain modelers to represent their preference for the plan quality that they are seek to optimise.

To explain the construction of the MOLP model for plan extraction in MetricLPRPG, the relaxed planning graph depicted in Figure 5.11 is taken as the example. All actions appearing between the first and goal layers in Figure 5.6 and 5.11 are taken as actions variables for the first objective function, $f_1(x)$. Assume that the plan metric stated in the problem is as follows,

```
(:metric minimise (+ (+ (* 3 (pollution)) (* 2 (resource-use)))
(* 0 (labour))))
```

This plan metric is taken as the second function, $f_2(y)$, the extra function which is introduced in MOLP model constructed in the solution development for MetricLPPRG. The constraints for these functions are actually constraints that are developed from achieving the relevant numeric fact that appears from the first until the goal layers. The overall MOLP model constructed for the problem discussed above can be seen as shown in Figure 5.12.

How can the MOLP model constructed in Figure 5.12 be used to obtain the solution? In MetricLLPRPG, the LP solver software is called to solve such constructed MOLPs. The solution obtained from the LP solver is employed to facilitate the selection of cheaper actions relative to the value of the plan metric during the relaxed plan extraction. The relaxed plan extraction for MetricLPRPG is accomplished according to Algorithm 1. This algorithm was also implemented in LPRPG [10] to extract the

Minimise $1.1ft + 1.1ftm + 1.1bs + 1.21bsm + 1.33sw + 1.46bh + 1.46bwh + 3pol + 2ruse$

subject to :

$tim' - tim - 3ftm - ft + sw = 0$
$wd' - wd - sw + bh + 2bhw = 0$
$lab' - lab - ft - bs - 2bsm = 0$
$pol' - pol - ftm = 0$
$ruse' - ruse - bs = 0$
$st' - st - bs + bh = 0$
$house' - house + bh + bhw = 0$

$house \geq 1$
$0 \leq ft \leq 4$
$0 \leq bs \leq 4$
$0 \leq ftm \leq 4$
$0 \leq bsm \leq 1$
$0 \leq bh \leq 4$
$0 \leq bwh \leq 4$

$0 \leq tim \leq 16$
$0 \leq st \leq 4$
$0 \leq wd \leq 12$
$0 \leq pol \leq 4$
$0 \leq ruse \leq 4$
wd, tim, pol, $rsue$, $house$, $st = 0$

Figure 5.12: LP Model for Plan Extraction in the Extended Settler Domain

relaxed plan. However, some modifications have been made in line 31 to represent the above constructed MOLP model, resulting in differences between LPRPG and MetricLPRPG solutions. The following discussion will elaborate the manual application of the above constructed MOLP in the algorithm.

In the first example, assume the value of the function represents the plan metric in the MOLP model constructed in Figure 5.12 is zero. After all, this is the best minimum value that the plan metric can hold. As a consequence of this, both *pollution* and *resource_use* variables are assigned the value zero. The *labour* variable, on the other hand, can hold any value since the coefficient for this variable is 0. With regard to these plan metric values, the MOLP model in Figure 5.12 is manually solved and possible values that can be assigned to each action variable are shown in column 2 in Table 5.2. The non zero values indicate the actions that can be selected in the constructed relaxed

plan. The assignment of non zero value to the action variables is stated in line 33 in the algorithm. Following this assignment, the plan length value obtained for this solution is 6. Since the plan cost value is 0, the value of the total sum of plan length and plan cost will be equivalent to value of the plan length. The extracted relaxed planning graph will consist of actions with the non zero value. For the solution developed in column 2, the constructed relaxed plan is shown in Figure 5.13. Similar to LPRPG, the actions that appear at the first layer will be selected as helpful actions for the next step in the search. This is indicated in line 35 in the algorithm.

```
0.001 : fell-timber location1
1.002 : fell-timber location1
2.003 : build-saw-mill location1
3.004 : fell-timber location 1
4.005 : fell-timber location1
5.006 : saw-wood location1
6.007 : saw-wood location1
7.008 : build-wood-house location1
```

Figure 5.13: Relaxed Plan Constructed for Example 1

Solving the MOLP model constructed in Figure 5.12 produces a single solution and therefore a single Pareto value. In the theory of weighted-sum technique, the weight given to the variable is changed several times in order to obtain a set of Pareto optimal values. As mentioned previously, the MOLP model constructed in MetricLPRPG heuristic consists of action and plan metric variables. The weight of each action variable is given according to the layer at which the action is currently attached. Therefore, these weights are fixed and cannot be changed in order to comply with the algorithm applied in the relaxed planning graph extraction. The weight of each plan metric variable is represented by the coefficient of such plan metric variables as stated by the domain modeler in the problem file. To demonstrate the consequence of changing the coefficient value of the plan metric variables, consider the following second example. In this example, assume the second function of MOLP in Figure 5.12 represents the following plan metric plan metric;

```
(:metric minimise (+ (+ (* 0 (pollution)) (* 0 (resource-use)))
(* 3 (labour))))
```

How do the above changes made in the plan metric affect the plan length and the total sum of the function as a whole? The new plan metric implies the modeler prefers plan that have cheaper *labour* cost than either *pollution* or *resource-use* cost.

With regard to the general constraints of the *labour* variable, as shown in Figure 5.7, the possible minimum value for this variable is 2. This is due to the fact that *build_saw_mill* action requires 2 units of *labour*. This action is necessary in any generated solution and unfortunately it does not have any alternative actions. To solve the MOLP , the most possible values that would be assigned to each action variable are presented in column 3 in Table 5.2. The non zero actions are selected in the constructed relaxed plan. The constructed relaxed plan for values assignment made in column 3 is shown in Figure 5.14.

```
0.001 : fell-timber-machine location1
1.002 : build-saw-mill location1
2.003 : fell-timber machine location1
3.004 : saw-wood location1
4.005 : saw-wood location1
5.006 : build-wood-house location1
```

Figure 5.14: Relaxed Plan Constructed For example 2

According to the manual variables assignments of the variables in column 3, the plan length found has decreased by 2 steps compared to the value of the relaxed plan length resulting from the variable assignment conducted in column 2. The plan cost is $0(2) + 0(0) + 3(2) = 6$. The total sum of plan cost and plan length is $6 + 6 = 12$. The heuristic value in MetricLPRPG is still based on the value of the relaxed plan length. Therefore, the heuristic value of the relaxed plan extracted from column 2 and 3 are 8 and 6. The two above examples have shown that the MOLP introduced in MetricLPRPG heuristic has successfully found the best actions with regard to the plan metric value.

Without considering the plan metric, the constructed relaxed plan would be different. As implemented in the LPRPG heuristic, the LP function only consists of action variables since the purpose of the heuristic is to minimise the value of the plan length. Therefore, the relevant objective function for the above problem can be stated as the following.

```
Minimise 1.1ft + 1.1ftm + 1.1bs + 1.21bsm + 1.33sw + 1.46bh +
1.46bwh
```

Assume, in the third example, the LP model only consists of the above objective function. To solve this LP function manually, the most possible values that can be assigned to each variable is shown in column 4 in Table 5.2. These assignments have resulted in the value of the relaxed plan length being 5, which is shorter than the length

of the relaxed plans constructed either from the values in column 2 or 3. The relaxed plan constructed based on these value assignments is shown in Figure 5.15. How much is the cost of this constructed relaxed plan? The plan cost is $3(1) + 2(1) + 0(3) = 5$, according to the plan metric function discussed in example 1 which is *3(pollution) + 2(resource-use) + 0(labour)*. Whereas, with regard to the plan metric in example 2, *0(pollution) + 0(resource-use) + 3 (labour)*, the plan cost is $0(1)+0(1)+3(3) = 9$. The plan cost is expensive with regard to the plan metric function in the first two examples though it has the shortest plan length.

```
0.001 : break-stone location1
1.002 : fell-timber machine location1
2.003 : build-sawmill location1
3.004 : saw-wood location1
4.005 : build-house location1
```

Figure 5.15: Relaxed Plan Constructed in LPRPG heuristic

5.4 Conclusion

MetricLPRPG has included the plan metric variables in its heuristic in order to take into account plan cost without neglecting the aspect of plan length. To accomplish this, MetricLPRPG has incorporated the Multi-objective Linear Programming(MOLP), in which the objective function of the MOLP is designed to minimise the total sum of plan length and cost simultaneously. Before doing the plan extraction, the upper and lower bounds of such plan metric variables are estimated through LP function which is similar to the bound estimations made for numeric precondition variables in the LPRPG planner. The implementation of MOLP in the MetricLPRPG heuristic has produced a significant improvement in the plan quality, whereby, the solutions constructed will support the selection of the cheaper actions with regard to the plan metric but within the achievable minimum value of plan length. Furthermore, as observed in the examples discussed in this chapter, the new heuristic implemented in MetricLPRPG is observed to be sensitive to any changes made in the plan metric variables. With this new heuristic, different solutions are developed for different plan metrics, though the problems solved have similar initial and goal states.

Algorithm 1: Relaxed Plan Extraction

```
Data: R - a metric RPG; PG - propositional goals;
      NG - numeric goals
Result: ha - helpful actions, h - A heuristic value
ha ← ∅, h ← 0;
q ← deepest-first priority queue of goal layers;
foreach p ∈ PG do
    l ← layer at which p first appears;
    insert (p, 1) into q[l].prop;
foreach f ∈ NG do
    l ← layer at which f first holds;
    insert (f, 1) into q[l].num;
while q not empty do
    (l, ⟨prop, num⟩) ← pop(q);
    foreach (p, w) ∈ prop do
        h ← h + w;
        a ← achiever for p;
        if a in action layer 0 then add a to ha;
        prop ← prop \ add effects of a;
        foreach propositional precondition pre of a do
            l ← layer at which pre first appears;
            if l > 0 then
                if ∃(pre, k) ∈ q[l].prop then
                    if k < w then k ← w;
                else insert (pre, w) into q[l].prop;
        ... similarly for numeric preconditions of a...;
    foreach non-linear (f, w) ∈ num do
        ... as in Metric-FF, modified for weights ...;
    foreach linear (f, w) ∈ num do
        foreach variable v used by f do
            LP' ← LP;
            if v is a positive metric RPG variable then
                LP' = LP' + {v = max(v_l)};
            else LP' = LP' + {v = min(v_l)};
            solve LP', minimising weighted action sum + plan metric sum;
            h ← h + w. LP' objective function value;
            av ← {action variable (a = c) ∈ LP' | c ≠ 0};
            foreach a ∈ av do
                if a is in layer-zero then add a to ha;
                foreach propositional precondition pre of a do
                    l ← layer at which pre first appears;
                    if ∃(pre, k) ∈ q[l].prop then
                        if k < w.c then k ← w.c;
                    else insert (pre, w.c) into q[l].prop;
```

Table 5.2: Manual Value Assignment of Action Variables

Action variables			
variable	**value 1**	**value 2**	**value3**
bs	0	0	1
ftm	0	2	1
ft	4	0	0
bsm	1	1	1
sw	2	2	1
bh	0	0	1
bwh	1	1	0
Plan metric variables			
pol	0	2	1
ruse	0	0	1
lab	6	2	3
plan length	*8*	*6*	*5*

Chapter 6

Results

6.1 Introduction

This chapter presents and analyses empirical results obtained from testing MetricLPRPG on a variety of domains and problems. The same set of domains and problems were used to evaluate its predecessor, LPRPG, and some other selected state-of-the-art numeric planners. The results obtained from these planners are compared against the ones produced by MetricLPRPG. The comparison is made based on the value of the *plan cost*.

The domains involved in the experiment are those described in Chapter 3: *Extended Settler*, *Bread*, *Production*, *Trader* and *Sugar*. These domains are semi-straightforward, according to the definitions given in Chapter 2. MetricLPRPG was specially designed for this kind of domain, which is characterised by the fact that the addition of any action to any plan will always increase its metric cost (or leave it unchanged), and never reduce it. However, shorter plans are not necessarily of lower cost since some choices of action for achieving a given effect can be much more expensive than others. These domains have a common structure in which they include choices of *consumer* and *producer* actions. Different action choices will affect different plan metric variables. Sometimes, the effect is on the values that similar variables can take. In consequence, several potential solutions with different plan cost and plan length can be generated. The detailed explanation on the above domains was discussed in Chapter 3.

6.2 Planners and Experiment Considerations

MetricLPRPG is a numeric planner that handles PDDL2.1. A number of state-of-art numeric planners have been chosen to compare their generated solutions against Met-

ricLPRPG. The selected state-of-art numeric planners involved in these experiments include LPRPG [10], Metric-FF [38], LPG-td [32] and MIPS-XXL [18]. These planners were developed by different researchers and have different capabilities. These planners are all categorised as numeric planners. They are all capable of handling the numeric features and plan metrics of PDDL2.1 [24]. The ability to handle the plan metric is important since the focus of the experiment is to examine the plan metric values produced in the solutions. Therefore, all the domains and problems involved in the experiment are encoded with PDDL2.1, and the necessary plan metric function is encoded in every problem file.

The aim of the experiment is to evaluate the plan "quality" in the results presented by the participating planners. Plan quality in this experiment refers to the plan cost value. However, conflicts often exist between the plan cost and plan length in domain-specific metric problems. This means that any attempt made to decrease the plan cost might increase the plan length value. Therefore, besides comparing the solutions based on the plan cost and plan length, the trade-off is examined through the percentage of the plan cost reduction/increment against the percentage of the plan length reduction/increment made in MetricLPRPG against the results obtained from other planners. Furthermore, as discussed in Chapter 5, the new heuristic implemented in MetricLPRPG is designed to deal with the trade-off that happens between plan length and plan cost. The automatic Validation tool for PDDL, which is called VAL [41], was used to obtain the plan cost of each generated solution. This comparison analysis is important to support the claim made for the new heuristic, that it produces different plans relative to the plan metric function encoded in the problem file. Furthermore, at the same time, the heuristic attempts to minimise the plan length while achieving the best value of the plan cost possible. In other words, the heuristic function seeks for an efficient minimum value of the total sum of plan cost and plan length, and achieve a better balance than other planners between the plan cost and plan length values.

In running the experiment, for each problem instance, the planners were limited to 1GB of memory and 1 hour of CPU time. This setup corresponds to the limits established by the International Planning Competition committee. There were 20 problem instances created for each domain. These problem instances are hand coded and can be accessed from the Strathclyde directory *apasshared*. All the planners were applied to the unmodified instances, without exception.

The plan quality issue is incorporated in the solution development in some numeric planners involved in the experiment, particularly, LPG-td and MIPS-XXL. Both planners are claimed to be able to produce improving quality plans. However, improve-

ment in the value of the plan metric is made through the generation of a sequence of plans, each of which is an improvement over the previous ones. In contrast, MetricLPRPG seeks good plan quality in the first generated plan. Moreover, the way that plan quality improvements are achieved is different between MIPS-XXL and LPG-td. In MIPS-XXL, a series of plans, in which each one improves the plan metric of the previous one, will be produced in a single execution. The searching for plan quality improvements is subjected to the availability of memory and CPU time given in the experiment setup. The last plan produced is the one that is taken for comparison. The latest plan usually represents the best plan quality so far that can be found within the given experiment setup. However, in LPG-td, the search for improvement in the plan quality is subject to the parameter setting in the execution. One of the required parameters represents the number of plans that should be generated in an execution. For example, if the value of the parameter is set to 3, the sequence of plans will consist of 3 plans. Normally, the third generated plan is expected to have the lowest plan cost compared to the value of the plan cost of the two plans that are generated previously, although many more plans might need to be produced to make a significant improvement. The number of plans needed to achieve the lowest plan cost is unknown. Since the user is free to set the parameter to suit the experimental setup, the parameter that represents the number of plans is set to 1 in every experiment conducted with LPG-td planner. In other words, only the first generated plan produced in LPG-td is taken into consideration in the comparison analysis.

6.2.1 Extended Settler Domain

The Extended Settler domain is a modified version of the Settlers domain in IPC3. The Settlers domain is considered a tough numeric domain in IPC3 since it is enriched with numeric interaction behavior. None of the planners that participated in IPC3 was able to solve all the problem instances. In the Extended Settler domain, another two *producer* and *consumer* actions such as *fell_timber_machine* and *build_wood_house* were added and these additional actions have increased the complexity of solution development. There were 20 new problem instances have been created for this experiment but still none of the planners was able to solve all of the problem instances. Each problem has different objective function. Each objective function consists of a different combination of the plan metric variables designed in the domain which entails; *labour*, *pollution* and *resource_use*.

Table 6.1: Results in the Extended Settler Domain

problem	**MetricLPRPG**		**LPRPG**		**Metric-FF**	
	length	value	length	value	length	value
p01.pddl	12	52	12	95	11	76
p02.pddl	18	0	14	9	12	12
p03.pddl	49	141	59	176	not solved	
p04.pddl	69	27	60	42	58	42
p05.pddl	77	0	75	17	65	21
p06.pddl	12	5	12	11	11	9
p07.pddl	12	1	12	2	11	2
p08.pddl	8	0	9	1	7	1
p09.pddl	9	0	9	1	7	1
p10.pddl	35	44	36	51	38	65
p11.pddl	61	0	51	25	52	34
p12.pddl	51	0	43	19	42	31
p13.pddl	54	44	54	51	57	70
p14.pddl	62	54	62	61	not solved	
p15.pddl	not solved		122	180	not solved	
p16.pddl	245	669	288	749	not solved	
p17.pddl	not solved		not solved		not solved	
p18.pddl	77	0	75	17	not solved	
p19.pddl	not solved		not solved		not solved	
p20.pddl	not solved		167	575	not solved	

The domain and problems were tested against the following three planners only: MetricLPRPG, LPRPG and Metric-FF. The other two planners, MIPS-XXL and LPG-td were excluded since they are unable to handle both *assign* and ADL statements that are encoded in the domain. The plan cost for each solution produced by the participating planners is shown in Table 6.1. The value 0 represents the value of the plan metric given in the problems. For example, the plan metric of the solution produced by MetricLPRPG for problem *p02.pddl* is zero. This value is extremely low compared to the solutions generated from both LPRPG and Metric-FF planners. The zero values of the plan metric are also obtained in the solutions produced by MetricLPRPG for problem instances *p05.pddl*, *p08.pddl*, *p09.pddl*, *p11.pddl*, *p12.pddl* and *p18.pddl*. These plan cost values result from choosing the alternative actions that have no effect on the value of the encoded plan metric function in the said problem instances. For example, if the plan metric in such a problem is to minimise the *pollution* value, the developed solution will include those actions that increase the value of other plan metric variables but do not increase pollution. As can be seen from Table 6.1, MetricLPRPG has

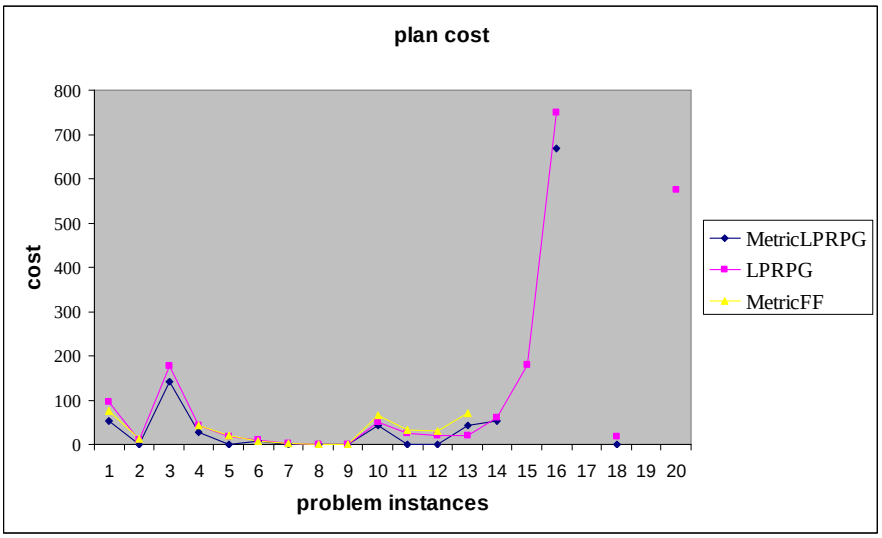

Figure 6.1: Plan Cost in Extended Settler Domain

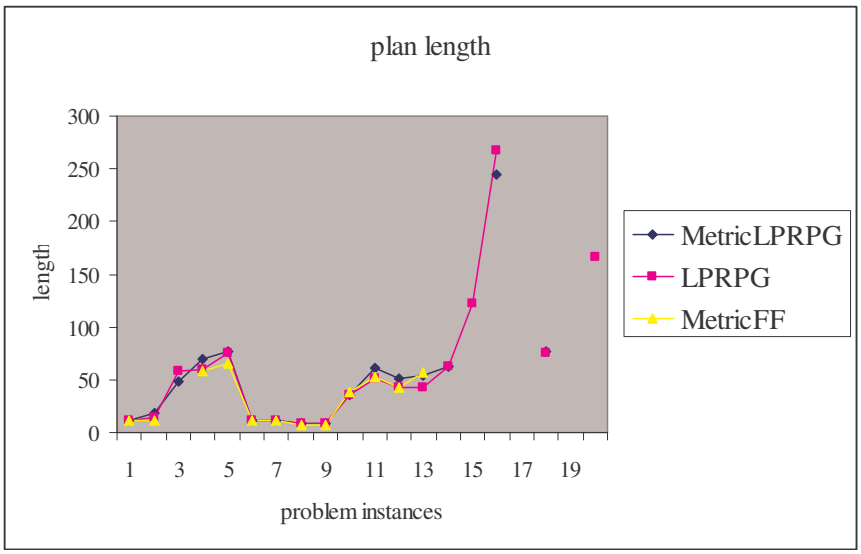

Figure 6.2: Plan Length in Extended Settler Domain

consistently given the minimum plan cost for all the developed solutions, in contrast to the plan cost value in the plans produced either by LPRPG and Metric-FF. The comparison of the plan cost is also depicted by the graph in Figure 6.1.

The plan length for the solutions generated by MetricLPRPG were generally longer than the plan length in the solutions produced by Metric-FF, except for plan length for

problems *p10.pddl* and *p13.pddl*. In these two problems, MetricLPRPG outperforms Metric-FF in both aspects of plan length and plan cost values. Interestingly, MetricLPRPG only increases the plan length in the solutions of the 6 problems, out of 16 problems that were also solved by both MetricLPRPG and LPRPG. Figure 6.2 presents the comparison of the plan length obtained from each planners in the form of graph.

Table 6.2: Percentage of Plan cost Reduction in MetricLPRPG in Extended Settlers domain

problem	**LPRPG**	**Metric-FF**
p01.pddl	45.26	31.58
p02.pddl	100	100
p03.pddl	19.89	not solved
p04.pddl	35.71	35.71
p05.pddl	100	100
p06.pddl	54.55	44.44
p07.pddl	50	50
p08.pddl	100	100
p09.pddl	100	100
p10.pddl	13.73	32.31
p11.pddl	100	100
p12.pddl	100	100
p13.pddl	13.73	37.14
p14.pddl	11.48	
p15.pddl		
p16.pddl	10.68	
p17.pddl		
p18.pddl	100	
p19.pddl		
p20.pddl		
average	**59.69**	**69.27**

The increment in the plan length is expected because the solution development in the domain-specific metric problems demonstrates the trade-off between plan length and plan cost values. Therefore, as has been mentioned earlier in many parts of this thesis, the heuristic developed in MetricLPRPG aims to minimise both plan length and plan cost values. In other words, MetricLPRPG attempts to reduce the plan cost whilst seeking a possible minimum value of the plan length. In order to support the above claim, the percentage of cost reductions given in the solutions developed by MetricLPRPG against the solutions developed by other participating planners are calculated. Table 6.2 shows the percentage of plan cost reduction in the plan generated

by MetricLPPRG against the plan cost produced by LPRPG and Metric-FF. In the first instance, MetricLPRPG achieved 45.26 % of the cost given by the plan generated by LPRPG and 31.58 % of the cost compared to plan produced by Metric-FF. Overall, on average MetricLPRPG achieved almost 60% of the plan cost given by the LPRPG planner and 69.27 % of the cost of the plans produced by Metric-FF.

Do the solutions generated from MetricLPRPG also exhibit as significant a change in the plan length as is demonstrated in the plan cost?. In other words, does a 60 % cost reduction also imply a 60% increment in the plan length value?. Table 6.3 shows the percentage of plan length increment or reduction obtained in the solution developed from MetricLPRPG against the solutions developed from other participating planners. On average, MetricLPRPG increases plan length by 2.58% compared to the plan length given by LPRPG and 15.26% compared to the plan length obtained by Metric-FF. These values are relatively small and insignificant compared to the benefit gained in cost reduction. Furthermore, these values do not imply the plan length always increases as the plan cost reduces. As can be seen, some problems demonstrate 0% and negative value of percentage in plan length increment. The 0% indicates the plan length obtained from both planners are similar though the plan cost value is difference. Whereas, the negative percentage implies the plan length produced from MetricLPRPG is shorter than the plan length produced by the comparison planner. Therefore, in such cases, MetricLPRPG outperforms its competitor in both aspects of plan length and plan cost. The values of percentage of cost reduction and plan length increment/decrement presented in Table 6.2 and Table 6.3 indicate the heuristic implemented in MetricLPRPG has successfully produced plans with reduced values of plan cost relative to the plan metric function.

6.2.2 Bread Domain

The bread domain is another producer-consumer domain constructed in this thesis. The plan metric encoded in problem instances designed for this domain contains *labour*, *pollution* and *energy* variables. The *producer* and *consumer* action choices encoded in this domain include *kneed-hand*, *kneed-machine*, *baking-oven* and *baking-charcoal*. The *kneed-hand* action increases the value of *labour*, *kneed-machine* and *baking-oven* increase the value of *energy*. Meanwhile, the action *baking-charcoal* increases the value of *pollution*. 20 problem instances have been created for this domain and then tested on all participating planners. The results obtained from the planners involved in the experiment are shown in Table 6.4. As can be seen, MetricLPRPG solved all the problem instances and consistently produced the minimum value of the plan cost in its

Table 6.3: Percentage of Plan length Increasing in MetricLPRPG in Extended Settler Domain

problem	**LPRPG**	**Metric-FF**
p01.pddl	0.00	9.09
p02.pddl	28.57	50.00
p03.pddl	-16.95	not solved
p04.pddl	15.00	18.97
p05.pddl	2.67	18.46
p06.pddl	0.00	9.09
p07.pddl	0.00	9.09
p08.pddl	-11.11	14.29
p09.pddl	0.00	28.57
p10.pddl	-2.78	-7.89
p11.pddl	19.61	17.31
p12.pddl	18.60	21.43
p13.pddl	0.00	-5.26
p14.pddl	0.00	
p15.pddl		
p16.pddl	-14.93	
p17.pddl		
p18.pddl	2.67	
p19.pddl		
p20.pddl		
average	**2.58**	**15.26**

generated plans compared to plan cost of the solutions produced by LPRPG, Metric-FF, LPG-td planners.

Table 6.4 shows that MetricLPRPG planner has produced solutions that have the plan values equal to its competitors. For example, in the solutions for problem instances started from *p02* until *p05*, the plan costs given by MetricLPRPG are similar to those produced by LPRPG planner. The number of such cases is small. Furthermore, the results are basically correlated to the plan metric encoded in the problems as well as the relationship between the plan metric variables and the actions. For example, generally in the above problem instances, the plan metric encoded in the problem is to minimise the cost of *labour* and *pollution*. Based on the implemented heuristic, MetricLPRPG will find alternative actions available in the domain that would not increase the values of the plan metric variables. As a result, the *kneed-machine* and *baking-oven* are chosen instead of *kneed-hand* and *baking-charcoal* actions in the solutions generated for above

Table 6.4: Results in the Bread Domain - Plan Cost

problem	MetricLPRPG	LPRPG	Metric-FF	LPG-td	MIPS-XXL
p01.pddl	11	12	11	19	10
p02.pddl	0	0	18	8	0
p03.pddl	6	6	10	9	0
p04.pddl	0	0	2	1	0
p05.pddl	5	5	27	62	0
p06.pddl	26	30	42	41	20
p07.pddl	11	23	23	30	11
p08.pddl	47	48	49	47	48
p09.pddl	3	not solved	19	26	0
p10.pddl	11	23	15	26	11
p11.pddl	22	40	38	32	30
p12.pddl	18	48	54	30	not solved
p13.pddl	24	30	52	59	not solved
p14.pddl	68	74	74	104	not solved
p15.pddl	34	34	34	78	not solved
p16.pddl	56	70	71	90	42
p17.pddl	58	102	97	72	not solved
p18.pddl	64	70	82	75	67
p19.pddl	88	97	111	101	not solved
p20.pddl	98	102	124	99	not solved

problem instances. These alternative actions as explained in Chapter 3 effect similar numeric variables, *dough* and *bread* or *bun*, but, increase the value of another plan metric variable called *energy* . See Appendix C to see the detailed domain description. Furthermore, the *kneed-machine* and *baking-oven*actions are also happen to be the actions that contribute to the minimum value of the plan length. Since the LPRPG planner attempts to produce a minimum plan length in the solution, it will pick these actions disregarding the plan metric encoded in the problem. The comparison of plan cost and plan length between these planners are depicted by the graphs in Figure 6.3 and Figure 6.4.

Besides generating similar plan cost values to those of its competitors, MetricLPRPG has been outperformed by MIPS-XXL planner in a few of the problem instances. MIPS-XXL produced plans with the lower plan cost compared to MetricLPRPG solutions in problem instances *p01.pddl*, *p03.pddl*, *p05.pddl*, *p06.pddl*, *p09.pddl* and *p16.pddl*. However, most of the MIPS-XXL solutions that have cheaper plan cost, have longer plan length compared to MetricLPRPG solutions. Moreover, the number of solutions

solved by MetricLPRPG is greater than the number of problem instances solved by the MIPS-XXL planner. This is can be seen from Table 6.5.

Table 6.5 presents the plan length value of the solutions generated by each of the planners. As expected, in many cases MetricLPRPG solutions have longer plan length values compared to other planner solutions. The best way to describe the improvement in the plan cost values against the draw back in the plan length made by MetricLPRPG is through the percentage values discussed above.

Table 6.5: Results in the Bread Domain - Plan length

problem	**MetricLPRPG**	**LPRPG**	**Metric-FF**	**LPG-td**	**MIPS-XXL**
p01.pddl	13	13	15	18	14
p02.pddl	20	20	22	20	20
p03.pddl	10	10	8	7	13
p04.pddl	10	10	8	10	7
p05.pddl	26	26	27	31	31
p06.pddl	31	26	27	35	31
p07.pddl	27	26	26	30	27
p08.pddl	39	39	39	39	39
p09.pddl	16	not solved	16	20	21
p10.pddl	21	20	20	24	21
p11.pddl	35	29	29	35	30
p12.pddl	71	50	44	22	not solved
p13.pddl	39	36	37	45	not solved
p14.pddl	47	44	44	53	not solved
p15.pddl	44	44	44	55	not solved
p16.pddl	38	39	34	41	34
p17.pddl	49	37	38	47	not solved
p18.pddl	47	46	46	51	46
p19.pddl	45	44	45	49	not solved
p20.pddl	60	61	60	61	not solved

Table 6.6 shows the percentage of plan cost reduction in the plan generated by MetricLPRPG against the plan cost produced by other planners. On average, MetricLPRPG produced plans with cost about 46% lower than the cost of the plan produced by LPG-td. MetricLPRPG has achieved up to 20% and 41% of the cost given in the plan produced from LPRPG and Metric-FF. But, the plan produced by MetricLPRPG on average is 18% higher than cost of the plan produced from MIPS-XXL. However, as mentioned previously, the number of problem solved by MetricLPRPG is higher than the number of problem solved by MIPS-XXL. Table 6.7 presents the percentage

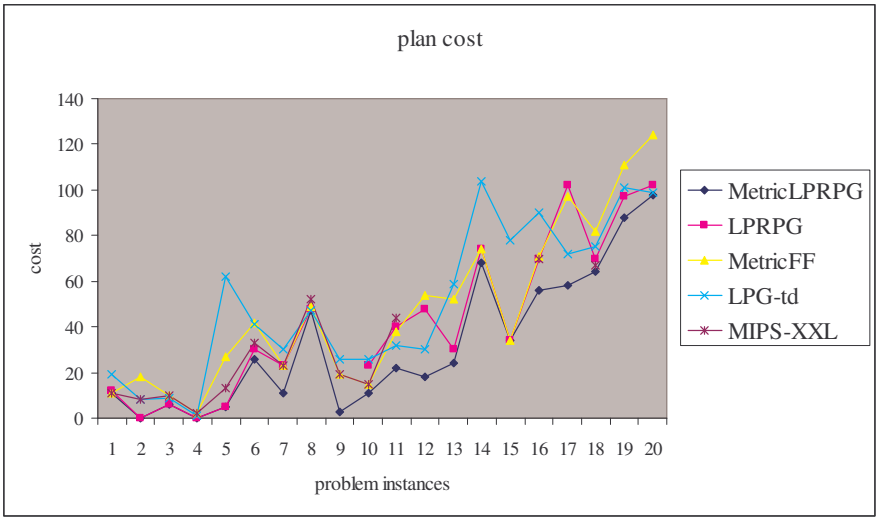

Figure 6.3: Plan Cost in Bread Domain

Figure 6.4: Plan Length in Bread Domain

of plan length increment in the solution produced by MetricLPRPG against the plan length values in the solutions produced by LPRPG, Metric-FF, LPG-td and MIPS-XXL. As can be seen the increment in the plan length is relatively small compared to the reduction made in plan cost.

Table 6.6: Percentage of cost reduction in MetricLPRPG vs other planner in Bread domain

problem	**LPRPG**	**Metric-FF**	**LPG-td**	**MIPS-XXL**
p01.pddl	8.33	0	42.11	-10.00
p02.pddl	0.0	100	100	0
p03.pddl	0.00	40	33.33	-100.00
p04.pddl	0.00	100.00	100.00	0.00
p05.pddl	0.00	81.48	91.94	0.00
p06.pddl	13.33	38.10	36.59	-30.00
p07.pddl	52.17	52.17	63.33	0.00
p08.pddl	2.08	4.08	0.00	2.08
p09.pddl		84.21	88.46	-100.00
p10.pddl	52.17	26.67	57.69	0.00
p11.pddl	72.50	42.11	31.25	26.67
p12.pddl	62.50	66.67	40.00	
p13.pddl	20.00	53.85	59.32	
p14.pddl	8.11	8.11	34.62	
p15.pddl	0.00	0.00	56.41	
p16.pddl	20.00	21.13	37.78	-33.33
p17.pddl	43.14	40.21	19.44	
p18.pddl	8.57	21.95	14.67	4.48
p19.pddl	9.28	20.727	12.87	
p20.pddl	3.92	20.97	1.01	
average	**19.80**	**41.12**	**46.04**	**-18.47**

6.2.3 Production Domain

This domain and its 20 problem instances were tested on all of the participating planners except MIPS-XXL. This is due to the inability of MIPS-XXL to solve any of the problem instances. The detailed explanation of this domain was given in Chapter 3. The objective function designed in each problem instance generally consists of the plan metric variables that have been previously encoded in the domain which include *hazard*, *machine-cost* and *labour*. However, each of the objective functions may either include different combinations of the plan metric variables or different coefficients value of the plan metric variables. The solutions developed by each planner are presented in Table 6.8. Although MetricLPRPG is found unable to produce any solution for problem *p03.pddl*, the solutions generated for other problem instances are considered competitive in terms of the plan cost value. MetricLPRPG has consistently given the lowest plan cost or at least the plan cost that is equal to the cost given by other planners. For example, the value of the plan cost for problem *p01* is 87 which is within the range 30%

Table 6.7: Percentage of length increment in MetricLPRPG vs other planners in Bread Domain

problem	**LPRPG**	**Metric-FF**	**LPG-td**	**MIPS-XXL**
p01.pddl	0.00	-13.33	-27.78	-7.14
p02.pddl	0.00	-9.09	0.00	0.00
p03.pddl	0.00	25.00	42.86	-23.08
p04.pddl	0.00	25.00	0.00	42.86
p05.pddl	0.00	-3.70	-16.13	-16.13
p06.pddl	19.23	14.81	-11.43	0.00
p07.pddl	3.85	3.35	-10.00	0.00
p08.pddl	0.00	0.00	0.00	0.00
p09.pddl		0.00	-20.00	-23.81
p10.pddl	5.00	5.00	-12.50	0.00
p11.pddl	20.69	20.69	0.00	16.67
p12.pddl	42.00	61.36	222.73	
p13.pddl	8.33	5.41	-13.33	
p14.pddl	6.82	6.82	-11.32	
p15.pddl	0.00	0.00	-20.00	
p16.pddl	-2.56	11.76	-7.32	11.76
p17.pddl	32.43	28.95	4.26	
p18.pddl	2.17	2.17	-7.84	2.17
p19.pddl	2.27	0.00	-8.16	
p20.pddl	-1.64	0.00	-1.64	
average	**7.29**	**9.23**	**5.12**	**0.25**

to 50% lower than the plan cost given in the solutions developed from LPRPG, Metric-FF and LPG-td. LPG-td, however, is observed to have outperformed all planners in the solution generated for problem *p02*. The plan generated for problem *p02* has the lowest plan cost as well as the shortest plan length.

On the other perspective of the plan quality, which is conventionally measured through the plan length, MetricLPRPG, as shown in Table 6.8 is found to have produced plans that have plan length longer than the plan length generated by LPRPG only in 8 problems. Overall, MetricLPRPG has produced longer plan lengths than those of the plans produced by Metric-FF and LPG-td. The graphs in Figure 6.5 and Figure 6.6 illustrate the comparison of the plan cost and plan length in the solutions produced by each planner in the experiment.

Table 6.9 presents the percentage of cost reduction obtained in the solutions developed by MetricLPRPG against the cost of the solutions given by LPRPG, Metric-FF

Table 6.8: Results in the Production Domain

problem	MetricLPRPG		LPRPG		Metric-FF		LPG-td	
	length	value	length	value	length	value	length	value
p01.pddl	69	87	56	144	56	170	54	120
p02.pddl	81	172	81	187	65	231	44	163
p03.pddl	not solved		82	150	85	200	88	140
p04.pddl	118	176	not solved		108	221	97	188
p05.pddl	116	120	81	120	112	230	92	120
p06.pddl	63	12	66	31	48	24	52	16
p07.pddl	63	147	65	151	56	163	59	159
p08.pddl	54	35	60	45	46	55	48	50
p09.pddl	63	147	68	156	56	183	67	165
p10.pddl	71	34	71	38	71	327	74	323
p11.pddl	77	42	71	62	71	47	71	67
p12.pddl	82	32	78	86	62	62	62	92
p13.pddl	8	10	2	20	2	20	4	20
p14.pddl	22	38	23	38	20	48	22	38
p15.pddl	75	292	65	306	58	316	59	318
p16.pddl	76	235	65	256	58	291	57	264
p17.pddl	85	98	86	194	75	158	88	194
p18.pddl	80	135	87	205	67	170	70	135
p19.pddl	66	48	50	76	44	88	56	102
p20.pddl	122	777	123	795	118	1141	118	963

and LPG-td. As can be seen, on average, MetricLPRPG produced plans with cost about 34% lower than the cost of the plans produced by Metric-FF. Moreover, these plans achieved up to 24% and 27% of the cost given in the plans produced from LPRPG and LPG-td respectively. The tremendous cost improvement is found exhibited in the problem *p06.pddl*. In this problem, MetricLPRPG generated a solution that cost 61% less than the solution obtained from LPRPG, and 25% to 50% less than the plan cost generated from both LPG-td and Metric-FF planners. The percentage of cost reduction would suggest that MetricLPRPG has made a significant improvement in terms of reducing the plan cost compared to the other participating planners.

Table 6.10 shows the percentage of plan length increment in the solutions produced by MetricLPRPG against the plan length values in the solutions produced by LPRPG, Metric-FF and LPG-td. The difference in the values of plan lengths between the solutions produced by MetricLPRPG and LPRPG is small. On average, MetricLPRPG produces plans 8% longer than the plans produced by LPRPG. The plan lengths was

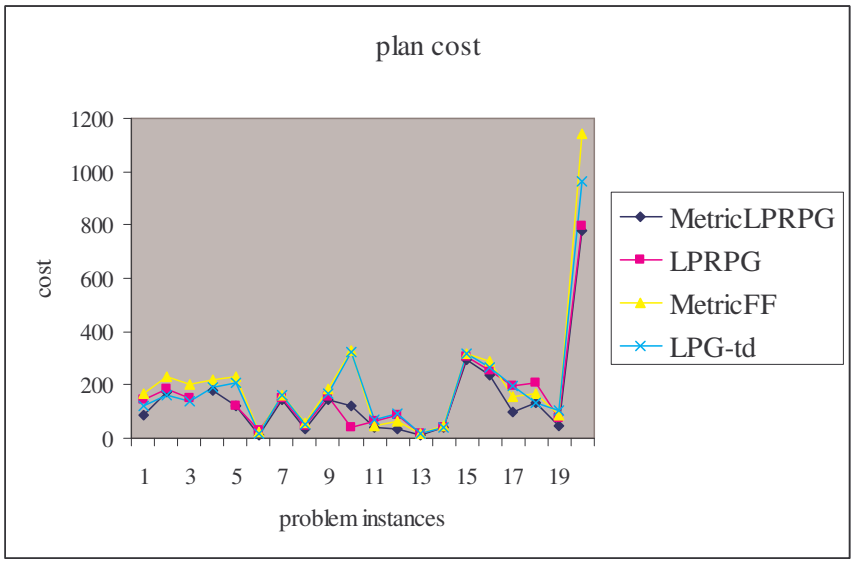

Figure 6.5: Plan Cost in Production domain

Figure 6.6: Plan Length in Production domain

Table 6.9: Percentage of cost reduction given by MetricLPRPG in the Production Domain

problem	**LPRPG**	**Metric-FF**	**LPG-td**
p01.pddl	39.5	48.8	27.5
p02.pddl	8.0	25.5	-5.5
p03.pddl			
p04.pddl		20.4	6.4
p05.pddl	0	47.8	0
p06.pddl	61.3	50	25
p07.pddl	2.6	9.8	7.5
p08.pddl	22.2	36.4	30
p09.pddl	5.8	19.7	10.9
p10.pddl	10.5	89.6	89.3
p11.pddl	32.3	10.6	37.3
p12.pddl	62.8	48.4	65.2
p13.pddl	50	50	50
p14.pddl	0	20.8	0
p15.pddl	4.6	7.6	8.2
p16.pddl	8.2	19.2	10.9
p17.pddl	49.5	37.9	49.5
p18.pddl	34.1	20.6	0
p19.pddl	36.8	45.5	52.9
p20.pddl	2.3	31.9	19.3
average	**23.9**	**33.7**	**27.2**

found to be about 18% longer than the plan lengths produced by Metric-FF, and about 15% longer than those of LPG-td. The increase in the plan lengths exhibited by MetricLPRPG is expected and can be considered small. In sum, these results also show that the heuristic obtained efficient values of the plan length and the plan cost.

6.2.4 Trader Domain

Apart from MetricLPRPG, there were only two other numeric planners capable of solving the Trader Domain. The other numeric planners were LPRPG and LPG-td. The numeric variables *liquidity*, *expenditure*, *resource*and *labour* designed in the domain were used as the plan metric variables in the problem instances. Similar to the domains discussed in the previous section, 20 problems instances were also created for the purpose of this experiment. Each of the problem instance is encoded with a plan metric function. In general, at the initial state, the *trader* is given an amount of *cash* and several *goods*. At the goal, the *trader* is expected to have some quantities of other

Table 6.10: Percentage of length increasing given by MetricLPRPG in the Production Domain

problem	**LPRPG**	**Metric-FF**	**LPG-td**
p01.pddl	18.8	18.8	21.7
p02.pddl	0	19.8	53.6
p03.pddl			
p04.pddl		8.5	30.4
p05.pddl	30.2	3.4	34.8
p06.pddl	-4.8	23.8	15.9
p07.pddl	-3.2	11.1	5.8
p08.pddl	11.1	14.8	8.7
p09.pddl	-7.9	11.1	-5.7
p10.pddl	0	0	-4.3
p11.pddl	7.8	7.8	8.7
p12.pddl	4.9.0	24.4	29.0
p13.pddl	75	75	5.8
p14.pddl	-4.5	9.1	0
p15.pddl	13.3	22.7	23.2
p16.pddl	14.5	23.7	27.5
p17.pddl	-1.2	11.8	-4.3
p18.pddl	-8.8	16.3	14.5
p19.pddl	24.2	33.3	14.5
p20.pddl	-0.8	3.3	5.8
average	**8**	**18**	**15**

goods that sell in the markets which are usually different from the ones that already on hand. The *trader* will travel to the potential markets and use available resources such as *cash* and *goods*, to obtain other *goods*. The 20 problem instances that have been created had different plan metric values in the initial and goal states. The choices of actions available in the domain to achieve the goal states were explained in Chapter 3.

The values of plan length and plan cost of the solutions developed by the participating planners are presented in Table 6.11. As can be seen, all planners were generally able to produce solutions to all problem instances. MetricLPRPG consistently generated the minimum plan cost solutions compared to LPRPG and LPG-td. In fact, the gap in plan cost value between the solutions from MetricLPRPG and LPG-td is considered big and significant. For example, the value of the plan cost in the solution for problem *p15.pddl* is *146* compared to plan cost value *1189* for the plan generated by LPG-td. Apart from that, the length of the solutions produced by MetricLPRPG is generally shorter than the solutions produced by LPG-td. The graphs in Figure 6.7 and

Figure 6.8 illustrate the comparison of plan cost and plan length between the planners that were involved in the experiment.

Table 6.11: Results in the Trader Domain

problem	**MetricLPRPG**		**LPRPG**		**LPG-td**	
	length	value	length	value	length	value
p01.pddl	35	16	38	23	144	65
p02.pddl	46	18	45	32	356	501
p03.pddl	42	12	37	54	130	160
p04.pddl	35	36	31	56	137	153
p05.pddl	37	20	57	29	689	146
p06.pddl	65	42	95	75	528	167
p07.pddl	136	88	71	220	313	383
p08.pddl	81	28	72	48	406	124
p09.pddl	141	10	not valid plan		440	266
p10.pddl	66	64	67	90	509	138
p11.pddl	47	28	44	68	256	160
p12.pddl	153	148	92	267	554	445
p13.pddl	131	70	70	254	896	375
p14.pddl	90	294	88	274	866	902
p15.pddl	133	146	82	288	977	1189
p16.pddl	138	22	84	122	882	178
p17.pddl	57	18	51	28	682	20
p18.pddl	59	32	51	50	225	36
p19.pddl	67	17	73	61	671	173
p20.pddl	91	80	67	128	448	550

The percentage of plan cost improvement made in the solutions produced by MetricLPRPG against the cost incurred in the solutions developed by LPRPG and LPG-td is shown in Table 6.12. The biggest improvement of plan cost has been made in MetricLPRPG against the values of the plan cost produced by LPG-td. On average, the plan cost of the solution produced by MetricLPRPG is 74% cheaper than the plan value produced by LPG-td. The solutions obtained from MetricLPRPG were also found to be about 46% cheaper than the plan cost in the solutions generated by LPRPG.

In terms of plan length, MetricLPRPG generally produced plans about 35% longer than the plans produced by LPRPG. Table 6.13 presents the percentage of increase in the length of plans produced by MetricLPRPG solutions against the solutions developed by LPRPG and LPG-td. However, in some problems, MetricLPRPG found shorter plans than those generated by LPRPG. This is indicated by the negative values of

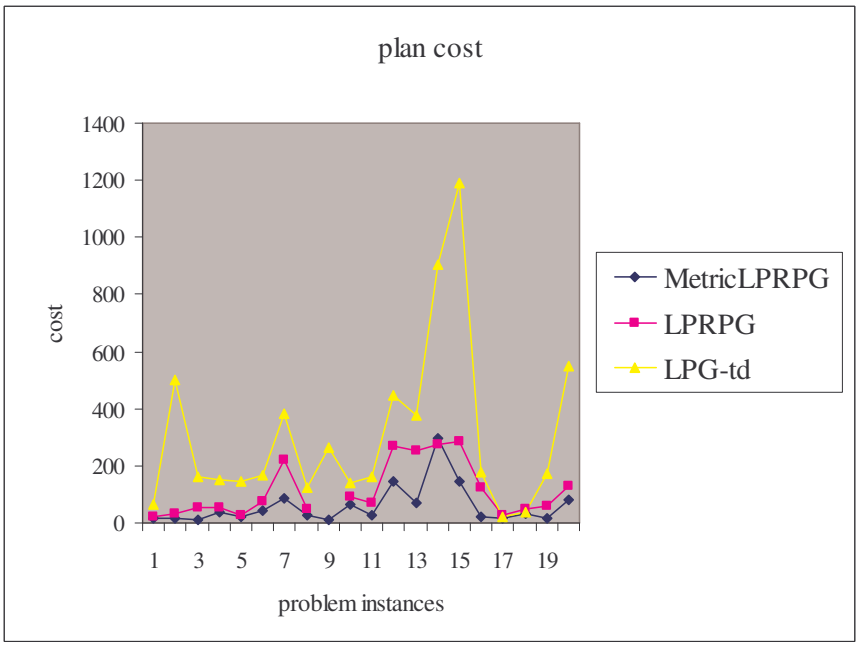

Figure 6.7: Plan Cost in Trader Domain

percentage in the table. In contrast, the lengths of solutions developed by MetrcLPRPG were always shorter than the lengths of solutions generated by LPG-td. As can be seen the percentage value is negative in all solutions. These results support the claim that the heuristic of MetricLPRPG simultaneously minimises the values of the plan length and plan cost.

6.2.5 Sugar Domain

The plan cost of the *Sugar* domain is determined by three plan metric variables: *inventory-cost*. *holding-cost* and *mill-cost*. These plan metric variables were attached to the different choices of action that achieve similar logical states. The detailed explanation on this was given in Chapter 3. The domain and its 20 problem instances were tested against MetricLPRPG, LPRPG, Metric-FF and LPG-td planners. MetricLPRPG has successfully solved all the problem instances. The values of the plan cost and plan length of the solutions developed from each of the planners are given in Table 6.14. The plan values in the table show that MetricLPRPG produced a plan with the lowest plan cost, in contrast to the solutions developed by other planners.

However, MetricLPRPG has produced a solution to problem *p15* which has the second worst metric value. On the other hand, as shown, Metric-FF has outperformed

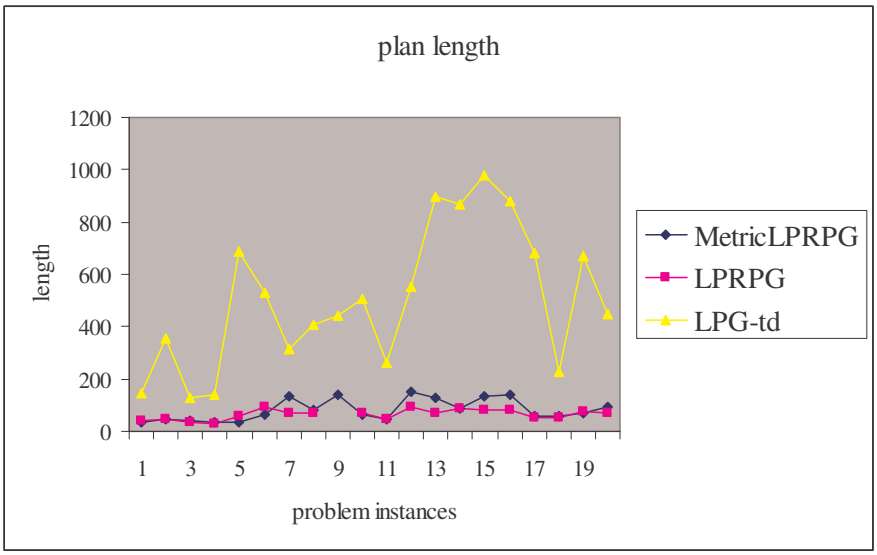

Figure 6.8: Plan Length in Trader Domain

other planners by producing a plan that has a plan cost of 120, which is the lowest value compared to the other solutions. What sort of circumstances contribute to these results? Let us closely examine the plan length values given by the other planners involved in the experiment. Although Metric-FF gives the minimum plan cost, the value of the plan length is 68, which is longer than the lengths of the solutions obtained by both MetricLPRPG and LPRPG. The values of the plan length and plan cost of the solution obtained by MetricLPRPG results from the heuristic which, as discussed, seeks to minimise aspects of both plan length and plan cost. It attempts to obtain the best plan cost for the best plan length that can be achieved.

Reduction in the plan cost value does not always greatly increase the plan length. Table 6.15 and Table 6.16 illustrate the percentage of cost reduction and length increment gained in MetricLPRPG solutions in contrast to the plans generated by other planners. For example, in the first problem instance MetricLPPRG produced a plan with cost 50% lower than LPRPG. Furthermore, this plan cost is 40 % and 71% lower than those solutions produced by Metric-FF and LPG-td respectively. The increase in the plan length is observed in MetricLPRPG solutions, however, the percentage

Table 6.12: Percentage of cost reduction in MetricLPRPG in Trader Domain

problem	**LPRPG**	**LPG-td**
p01.pddl	30.43	75.38
p02.pddl	43.75	96.41
p03.pddl	77.78	92.50
p04.pddl	35.71	76.47
p05.pddl	31.03	86.30
p06.pddl	44.00	74.85
p07.pddl	60.00	77.02
p08.pddl	41.67	77.42
p09.pddl		96.24
p10.pddl	28.89	53.62
p11.pddl	58.82	82.50
p12.pddl	44.57	66.74
p13.pddl	72.44	81.33
p14.pddl	-7.3	67.4
p15.pddl	49.3	87.7
p16.pddl	81.9	87.6
p17.pddl	35.7	10.0
p18.pddl	36.0	11.1
p19.pddl	72.1	90.2
p20.pddl	37.5	85.5
average	**46**	**74**

increase in length is much lower than the percentage reduction in the cost. Another interesting cost reduction made by MetricLPRPG appears in the plan generated as a solution to problem *p09.pddl*. In this problem, MetricLPRPG has made more than 70% cost saving in the plan cost value in contrast to the plan cost values obtained from the plans generated by LPRPG, Metric-FF and LPG-td. On top of the plan cost values, the plan is shorter than the ones produced by LPRPG. Besides this problem instance, there are numerous problem instances that better combine plan length and plan cost qualities.

6.3 Conclusion

The heuristic designed in MetricLPRPG strives to minimise the plan cost with respect to the plan metric function encoded in the problem. This plan cost is minimised together with the plan length values during the development of such solution. In sum, the solution generated using this heuristic can be expected to exhibit a better balance

Table 6.13: Percentage of plan length increasing in MetricLPRPG compared to other planners in the Trader Domain

problem	**LPRPG**	**LPG-td**
p01.pddl	-7.9	-75.7
p02.pddl	2.2	-87.1
p03.pddl	13.5	-67.7
p04.pddl	12.9	-74.5
p05.pddl	-35.1	-94.6
p06.pddl	-31.6	-87.7
p07.pddl	91.6	-56.6
p08.pddl	12.5	-80.1
p09.pddl		- 68
p10.pddl	-1.5	-87.0
p11.pddl	6.8	-81.6
p12.pddl	66.3	-72.4
p13.pddl	87.1	-85.4
p14.pddl	2.27	-89.6
p15.pddl	62.2	-86.4
p16.pddl	64.3	-84.4
p17.pddl	11.8	-91.6
p18.pddl	15.7	-73.8
p19.pddl	-8.2	-90.0
p20.pddl	35.8	-79.7
average	**35**	**-81**

between the plan cost and plan length values than is achieved by other current planners.

A series of experiments has been conducted and the results obtained have shown that MetricLPRPG is a competitive planner, giving high quality solutions with respect to plan cost and plan length for the domain-specific metric problems. Most of the solutions given by MetricLPRPG are generally cheaper in terms of plan cost but slightly longer in terms of plan length. These results are expected and represent the behavior that often exists in domains in the domain-specific metric class. However, some of the solutions produced by MetricLPRPG are observed to have cheaper plan cost and shorter plan length compared to other solutions. This indicates that the heuristic is sensitive to the coefficient values of each of the plan metric variables. The coefficient values represent the weight given to each such metric variable in evaluating the goodness of a solution.

Table 6.14: Results in the Sugar Domain

problem	**MetricLPRPG**		**LPRPG**		**Metric-FF**		**LPG-td**	
	length	value	length	value	length	value	length	value
p01.pddl	20	18	17	36	17	30	32	63
p02.pddl	26	135	28	135	not solve		244	297
p03.pddl	25	42	21	57	33	42	156	159
p04.pddl	33	68	31	70	47	70	not valid plan	
p05.pddl	23	58	25	58	not solve		94	93
p06.pddl	35	57	29	140	not solve		114	200
p07.pddl	11	54	11	54	not solve		12	54
p08.pddl	15	114	23	134	19	114	36	113
p09.pddl	44	34	140	132	30	120	103	160
p10.pddl	50	63	32	159	55	109	92	195
p11.pddl	21	14	16	21	11	35	14	20
p12.pddl	36	18	not solve		not solve		118	18
p13.pddl	20	73	18	200	17	117	19	117
p14.pddl	21	35	19	35	not solve		162	27
p15.pddl	26	184	36	180	68	120	96	212
p16.pddl	24	125	24	125	not solve		82	120
p17.pddl	21	46	not solve		not solve		42	76
p18.pddl	19	120	not solve		20	120	21	120
p19.pddl	50	350	54	375	76	400	222	415
p20.pddl	37	150	not solve		fail plan		175	162

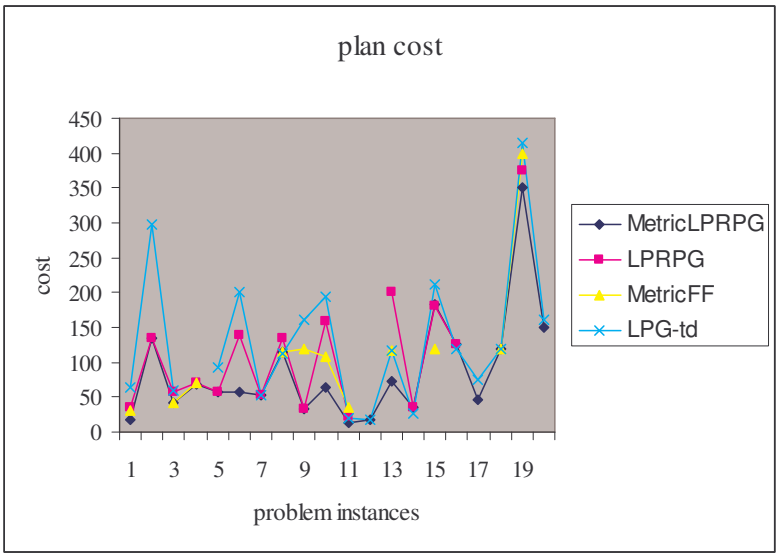

Figure 6.9: Plan Cost in Sugar Domain

Table 6.15: Percentage of Cost reduction in MetricLPRPG against other planners in Sugar Domain

problem	**LPRPG**	**Metric-FF**	**LPG-td**
p01.pddl	50	40	71.4
p02.pddl	0		54.5
p03.pddl	26.3	0	73.6
p04.pddl	2.9	2.9	
p05.pddl	0		37.6
p06.pddl	59.3		71.5
p07.pddl	0		0
p08.pddl	14.9	0	-0.8
p09.pddl	74.2	71.7	78.8
p10.pddl	60.4	42.0	67.7
p11.pddl	33.3	60	30
p12.pddl			0
p13.pddl	63.5	37.6	37.6
p14.pddl	0		-29.6
p15.pddl	-2.2	-53.3	13.2
p16.pddl	0		-4.2
p17.pddl			39.5
p18.pddl		0	0
p19.pddl	6.7	12.5	15.7
p20.pddl			7.4
average	**24.3**	**19.4**	**29.7**

Table 6.16: Percentage of length increment in MetricLPRPG against other planners in Sugar Domain

problem	**LPRPG**	**Metric-FF**	**LPG-td**
p01.pddl	17.6	17.6	-37.5
p02.pddl	-7.1		-89.3
p03.pddl	19.0	-24.2	-84.0
p04.pddl	6.5	-29.8	
p05.pddl	-8.0		-75.5
p06.pddl	20.7		-69.3
p07.pddl	0.0		-8.3
p08.pddl	-34.8	-21.1	-58.3
p09.pddl	-68.6	46.7	-57.3
p10.pddl	56.3	-9.1	-45.7
p11.pddl	31.3	90.9	50.0
p12.pddl			-69.5
p13.pddl	11.0	17.6	5.3
p14.pddl	10.5		-87.0
p15.pddl	-27.8	-61.8	-72.9
p16.pddl	0.0		-70.7
p17.pddl			-50.0
p18.pddl		-5.0	-9.5
p19.pddl	-7.4	-34.2	-77.5
p20.pddl			-78.9
average	**1.2**	**-1.1**	**-51.9**

Chapter 7

Conclusion

7.1 Summary

Numeric planning allows resources to be efficiently modelled in the planning domain. Optimisation over the metric resources defined in the planning domain is achieved through a plan metric, a new field introduced in PDDL2.1. The plan metric provides a means whereby solutions developed for particular problems can be evaluated based on the plan cost or the value of the plan metric. Furthermore, several solutions with different plan costs relative to the value of the plan metric have the potential to be generated for problems that have similar initial and goal states. These features are exhibited in a problem class called the *domain-specific metric problem* class.

The domain-specific metric problem class is a class of planning problems in which the plan metric is used to direct search towards high quality solutions. This problem class basically combines aspects of both planning and scheduling. As a result, the problem structure is blended with *how* action choices together with *what* resource choices. In the development of its solution, seeking for resource optimisation is as important as finding a feasible solution. Finding the minimum number of actions does not guarantee the optimal value of the plan metric. In other words, a conflict between minimising the number of actions and optimising the value of the plan metric is normally presented while seeking for the solution of such problem. This feature is interesting and poses a new challenge to the planning community. Furthermore, this feature is often demonstrated in many real world applications, but, it has been largely ignored by the community. Attempts to solve this problem class will increase the applicability of planning technology to real world problems

The thesis has conducted an analysis on the role of the plan metric in the metric domains benchmarked in the planning competition series. Four types of plan met-

ric are defined. They are: (1) *strictly-straightforward*, (2) *straightforward*, (3) *semi-straightforward* and (4) *expressive*. In general, the solution development of the planning problem with the plan metric defined as in definitions (3) and (4) is more subtle than the plan metric in definitions (1) and (2). The majority of the numeric domains in the IPCs, with the exception of *Settlers*, were found to encode the plan metric as either *strictly-straightforward* or *straightforward*. In these two types of plan metric encoding, the minimum value of the metric function is obtained by minimising the number of actions in the plan. This indicates that plan metrics that combine reasoning over several resources are not yet much in use in the benchmark domains.

Most of the numeric planners, even if they use PDDL2.1, focus on finding shortest plans length and the plan quality is considered only to the extent that it coincides with optimal length. In other words, the minimum plan length also result in the minimum value of the plan metric. Some numeric planners improve the plan quality by producing a sequence of plans in which each successor is an improvement over the predecessor. In this approach, the improvement is made by choosing actions with the minimum cost.

MetricLPRPG the extended planner developed in this thesis extends LPRPG by incorporating the plan metric in its heuristic. The detailed discussion on the development of this new heuristic was given in Chapter 5. The MetricLPRPG planner is specially designed to handle the trade-offs that occur in solving problems in the *domain-specific metric* problem class. Its heuristic attempts to minimise the value of plan cost and plan length simultaneously during the relaxed plan extraction phase. This is accomplished through modelling the heuristic using Multi-objective linear programming. Multi-objective linear programming is a widely used optimisation tool in the operational research community to balance trade-offs between different objectives in multiple objective functions problems. The LP solution is used to facilitate actions selection in which the actions with cheaper cost according to the plan metric will be chosen in the constructed relaxed plan. The results obtained in Chapter 6 show the new heuristic is competitive in providing a better value of plan quality particularly if the plan metric is encoded as semi-straightforward. Instead of giving a better value of the plan metric, the heuristic also produces a competitive value of the plan length. Although the increment in the value of the plan length is expected and presented in almost all the developed solutions, on average, its percentage is found to be lower than the percentage of the improvement made in the plan metric. This analysis is comprehensively discussed in Chapter 6.

The contributions of this work to the community can be summarised into two categories. The first contribution relates to the exploration of the new planning problem

class called the domain-specific metric problem class. This thesis has undertaken an in depth study on how the plan metric encoded in the domain-specific metric can result in the trade-off between plan length and plan cost in it solution development. Furthermore, the thesis has constructed a new classification of the plan metric encoding according to four definitions: 1) **strictly-straightforward**, 2) **straightforward**, 3) **semi-straightforward** and 4) **expressive**. This thesis focuses attention on the metrics described as **semi-straightforward**. These classifications furthermore, can facilitate approaches that can be taken in the development of solutions to such problems.

The second contribution is the novel heuristic that includes the plan metric variables encoded in the problem together with the plan length. The solution for the constructed multi-objective linear programming model is aimed at minimising the value of the total sum of the plan metric and plan length. Through this, the new heuristic attempts to obtain a better balance between the two parameters; plan length and plan cost and guides the selection of the minimum cost actions in the relaxed plan. This heuristic has been successfully implemented in the extended planner written in this thesis.

7.2 Limitation

MetricLPRPG has performed competitively in the domain-specific metric problems constructed in the thesis. In many cases it has successfully produced solutions that have a minimum value of the plan metric compared to its competitors. However, these values are not guaranteed to be optimal in terms of either plan length or plan cost alone.

The reason for this is that the quality obtained in the solution is connected to the approach applied in solving the multi-objective function. As described in Chapter 5, a simple weighted sum technique was applied to solve the constructed multi-objective function. According to this technique, a weight is given to each variable in the constructed objective function. The value of the weight is highly correlated to the knowledge of the modeller on the problem. In addition, these weights must be changed several time to obtain a set of efficient solutions. However in its implementation in MetricLPRPG, the weights are restricted to certain values and set only once in the development of the solution. This implies only a single relaxed plan is obtained for an evaluated state, s. There is not much that can be done for the weight given to the action variables. The current value of the weight given to each action variable reflects the way that helpful actions are chosen in the relaxed plan algorithm. Changes made

to these values might result in infeasible solutions, whereas the weight given to each plan metric variable in the objective function is directly taken from the coefficient of the plan metric variables set in the problem. This approach reflects the knowledge of the domain modeller about the problem. By having all these weights the problem is solved and result in a single solution and not a set of solutions as expected in typical applications of the weighted sum approach. This approach cannot guarantee that the most efficient solution has been achieved in the construction of such a solution. Furthermore, the direct mapping of the coefficient to the weight does not always retain the relative importance of the components of the plan metric.

The structure of the current domain-specific metric is very rigid. In order to get the benefit of the new heuristic, behavior that influences the plan metric must be encoded by means of so-called *producer* and *consumer* action choices. This is due to the fact that the multi-objective function is only constructed for numeric facts in which it only minimises the cost and number of actions for achieving such numeric facts. In other words, the multi-objective function is not constructed for propositional or logical facts in the domain. Therefore, it will not minimise the value of the plan metric in achieving any particular logical facts. As a result, the domain encoding becomes unrealistic since the alternative resources that are usually affected by different plan metric variable must be encoded in the actions that achieve the numeric part.

For example, consider a domain in which several alternative vehicles can be used to transport materials from one location to another. These vehicles have different capacities and cost. This means, transporting certain goods from one location to another using different vehicles should incur different costs. Generally, activities involved in moving goods from one location to another can be accomplished by two separate different actions. First, loading the goods into the selected vehicle. Second, moving the loaded vehicle to other location. These two actions basically combine both logical and numeric facts. The *load* action achieves the numeric fact whereas the *move* action achieves the logical part. Since the *move* action normally updates the propositional part, it will not be included in the heuristic. In order to make it work for the implemented heuristic, several load actions, each for different types of vehicle, might need to be encoded in the domain. These actions represent the alternative consumer actions that can be used in the constructed relaxed plan. For example, if the choice of vehicles available in the problem consists of *truck* and *cart*. Therefore, actions such as *load_truck* and *load_cart* have to be encoded in the domain. The metric cost has to be updated every time an item is being loaded into these vehicles. This can lead to an inefficient domain encoding. Furthermore, the cost of using a vehicle should be incurred at the

point where the decision to use a particular vehicle is made. However, since this kind of action only includes propositions, it is currently not being considered by the heuristic.

7.3 Future Work

The heuristic exploited in the current implementation applies the weighted-sum technique to obtain the solution, as explained in Chapter 5. However, it only considers a static weight. It means that the coefficients of the plan metric variables, given by the domain modeller are simply taken and mapped to the weights attached to each plan metric variable in the objective function. This approach results in a single solution or only one Pareto optimal value. If the domain designer attempts to find another optimal solution, although for similar preference of the metric variables, it is necessary to solve the same problem with many different coefficients and repeatedly apply the planner to the problem. With this approach, the designer is still confronted with choosing the most appropriate solution based on intuition.

The first direction of the future work is to change the weight given to each plan metric variable dynamically and automatically. This means an appropriate algorithm is constructed in the planner so that it changes the value of the weights. However, any changes made to the weight is still constrained by the weight given by the domain modeller as stated in the problem. This approach can be thought of as similar to those approaches taken by numeric planners that improve plan quality through a sequence of plans. Instead of improving the estimated action cost in the plan sequences, the future algorithm could consider a constant multiplier to be used as a basis to dynamically change the value of the weight. This constant multiplier scales the objective properly relative to the initial preference given by the domain modeller. With this, several plans, equivalent to a Pareto set, can be obtained. More sophisticated and efficient techniques could be applied to choose the optimal solution within the several generated solutions.

The current structure of the domain-specific metric is restricted. Another direction of the future work is to extend the resource choice to the propositional and temporal parts of the planning problem. It can be thought of as including different plan metric variables or different values that the variables can take to the different action choices that achieve similar propositional and temporal preconditions. The extension in the domain structure will perhaps increase the flexibility and allow real world problems to be modelled. In consequence, the heuristic is supposed to be modified according to the new domain structure. The implementation of the multi-objective function is extended to achieve such propositional and temporal facts. The number of action variables in

the objective function is expected to be increased since more actions will be included. Furthermore, it might also arise that conflicting factors in the objective function require more sophisticated techniques to resolve them.

Finally, the future work could consider the numeric problem with the plan metric as defined in Definition 4 in Chapter 2, *expressive metric function*. This is another type of metric function problem that is also exhibited in real world applications. The approach taken to minimise the number of actions together with the action cost might not be appropriate when there exist actions that reduce the value of the plan metric. Longer plans could have lower plan cost if the chosen actions can have negative cost. This problem is interesting and poses another new challenge to the planning community. The solution approach is expected to require more complicated techniques.

Appendices

Appendix A

Extended Settler Domain

```
(define (domain civ)
  (:requirements :fluents :typing :conditional-effects)
  (:types place vehicle - store
  resource)
  (:predicates
   (connected-by-land ?p1 - place ?p2 - place)
   (connected-by-rail ?p1 - place ?p2 - place)
   (connected-by-sea ?p1 - place ?p2 - place)
   (woodland ?p - place)
   (mountain ?p - place)
   (metalliferous ?p - place)
   (by-coast ?p - place)

   (has-cabin ?p - place)
   (has-coal-stack ?p - place)
   (has-quarry ?p - place)
   (has-mine ?p - place)
   (has-sawmill ?p - place)
   (has-ironworks ?p - place)
   (has-docks ?p - place)
   (has-wharf ?p - place)
   (is-cart ?v - vehicle)
   (is-train ?v - vehicle)
   (is-ship ?v - vehicle)
   (is-at ?v - vehicle ?p - place)
```

```
   (potential ?v - vehicle)
   )
  (:functions
   (available ?r - resource ?s - store)
   (space-in ?v - vehicle)
(labour)
(resource-use)
(pollution)
(housing ?p - place)
   )
  (:constants timber wood coal stone iron ore - resource)


  ;; A.1: Loading and unloading.

  (:action load
   :parameters (?v - vehicle ?p - place ?r - resource)
   :precondition (and (is-at ?v ?p)
      (> (available ?r ?p) 0)
      (> (space-in ?v) 0))
   :effect (and (decrease (space-in ?v) 1)
(increase (available ?r ?v) 1)
(decrease (available ?r ?p) 1)
(increase (labour) 1)))

  (:action unload
   :parameters (?v - vehicle ?p - place ?r - resource)
   :precondition (and (is-at ?v ?p)
      (> (available ?r ?v) 0))
   :effect (and (increase (space-in ?v) 1)
(decrease (available ?r ?v) 1)
(increase (available ?r ?p) 1)
(increase (labour) 1)))

  ;; A.2: Moving vehicles.
  ;; Moving trains and ships consumes coal, which has to be
  ;; loaded in the vehicle.
```

```
  (:action move-cart
   :parameters (?v - vehicle ?p1 - place ?p2 - place)
   :precondition (and (is-cart ?v)
      (connected-by-land ?p1 ?p2)
      (is-at ?v ?p1))
   :effect (and (not (is-at ?v ?p1))
(is-at ?v ?p2)
(increase (labour) 2)))

  (:action move-train
   :parameters (?v - vehicle ?p1 - place ?p2 - place)
   :precondition (and (is-train ?v)
      (connected-by-rail ?p1 ?p2)
      (is-at ?v ?p1)
      (>= (available coal ?v) 1))
   :effect (and (not (is-at ?v ?p1))
(is-at ?v ?p2)
(decrease (available coal ?v) 1)
(increase (pollution) 1)
))

  (:action move-ship
   :parameters (?v - vehicle ?p1 - place ?p2 - place)
   :precondition (and (is-ship ?v)
      (connected-by-sea ?p1 ?p2)
      (is-at ?v ?p1)
      (>= (available coal ?v) 2))
   :effect (and (not (is-at ?v ?p1))
(is-at ?v ?p2)
(decrease (available coal ?v) 2)
(increase (pollution) 2)
))

  ;; B.1: Building structures.

  (:action build-cabin
```

```
   :parameters (?p - place)
   :precondition (woodland ?p)
   :effect (and (increase (labour) 1) (has-cabin ?p)) )

  (:action build-quarry
   :parameters (?p - place)
   :precondition (mountain ?p)
   :effect (and (increase (labour) 2) (has-quarry ?p)))

  (:action build-coal-stack
   :parameters (?p - place)
   :precondition (>= (available timber ?p) 1)
   :effect (and (increase (labour) 2)
(decrease (available timber ?p) 1)
(has-coal-stack ?p)))

  (:action build-sawmill
   :parameters (?p - place)
   :precondition (>= (available timber ?p) 2)
   :effect (and (increase (labour) 2)
(decrease (available timber ?p) 2)
(has-sawmill ?p)))

  (:action build-mine
   :parameters (?p - place)
   :precondition (and (metalliferous ?p)
      (>= (available wood ?p) 2))
   :effect (and (increase (labour) 3)
(decrease (available wood ?p) 2)
(has-mine ?p)))

  (:action build-ironworks
   :parameters (?p - place)
   :precondition (and (>= (available stone ?p) 2)
      (>= (available wood ?p) 2))
   :effect (and (increase (labour) 3)
(decrease (available stone ?p) 2)
```

```
(decrease (available wood ?p) 2)
(has-ironworks ?p)))

  (:action build-docks
   :parameters (?p - place)
   :precondition (and (by-coast ?p)
      (>= (available stone ?p) 2)
      (>= (available wood ?p) 2))
   :effect (and (decrease (available stone ?p) 2)
(decrease (available wood ?p) 2)
(increase (labour) 2)
(has-docks ?p)))

  (:action build-wharf
   :parameters (?p - place)
   :precondition (and (has-docks ?p)
      (>= (available stone ?p) 2)
      (>= (available iron ?p) 2))
   :effect (and (decrease (available stone ?p) 2)
(decrease (available iron ?p) 2)
(increase (labour) 2)
(has-wharf ?p)))

  (:action build-rail
   :parameters (?p1 - place ?p2 - place)
   :precondition (and (connected-by-land ?p1 ?p2)
      (>= (available wood ?p1) 1)
      (>= (available iron ?p1) 1))
   :effect (and (decrease (available wood ?p1) 1)
(decrease (available iron ?p1) 1)
(increase (labour) 2)
(connected-by-rail ?p1 ?p2)))

  (:action build-house
   :parameters (?p - place)
   :precondition (and (>= (available wood ?p) 1)
(>= (available stone ?p) 1)
```

```
                )
   :effect (and (increase (housing ?p) 1)
(decrease (available wood ?p) 1)
(decrease (available stone ?p) 1))
  )

 (:action build-wood-house
  :parameters (?p - place)
  :precondition (and (>= (available wood ?p) 2)
)
   :effect (and (increase (housing ?p) 1)
(decrease (available wood ?p) 2)
   )
 )
  ;; B.2: Building vehicles.

  (:action build-cart
   :parameters (?p - place ?v - vehicle)
   :precondition (and (>= (available timber ?p) 1) (potential ?v))
   :effect (and (decrease (available timber ?p) 1)
(is-at ?v ?p)
(is-cart ?v)
(not (potential ?v))
(assign (space-in ?v) 1)
(forall (?r - resource) (and (assign (available ?r ?v) 0)))
(increase (labour) 1)
          )
  )

  (:action build-train
   :parameters (?p - place ?v - vehicle)
   :precondition (and (potential ?v) (>= (available iron ?p) 2))
   :effect (and (decrease (available iron ?p) 2)
(is-at ?v ?p)
(is-train ?v)
(not (potential ?v))
(assign (space-in ?v) 5)
```

```
                (forall (?r - resource) (and (assign (available ?r ?v) 0))
(increase (labour) 2)
           )
  )

  (:action build-ship
   :parameters (?p - place ?v - vehicle)
   :precondition (and (potential ?v) (>= (available iron ?p) 4))
   :effect (and (has-wharf ?p)
(decrease (available iron ?p) 4)
(is-at ?v ?p)
(is-ship ?v)
(not (potential ?v))
(assign (space-in ?v) 10)
                (forall (?r - resource) (and (assign (available ?r ?v) 0))
(increase (labour) 3)
           )
   )

  ;; C.1: Obtaining raw resources.

  (:action fell-timber
   :parameters (?p - place)
   :precondition (has-cabin ?p)
   :effect (and (increase (available timber ?p) 1)
(increase (labour) 1))
   )

  (:action fell-timber-machine
   :parameters (?p - place)
   :precondition (has-cabin ?p)
   :effect (and (increase (available timber ?p )3)
        (increase (pollution)1)
           )
  )

  (:action break-stone
```

```
   :parameters (?p - place)
   :precondition (has-quarry ?p)
   :effect (and (increase (available stone ?p) 1)
(increase (labour) 1)
(increase (resource-use) 1)
))

  (:action mine-ore
   :parameters (?p - place)
   :precondition (has-mine ?p)
   :effect (and (increase (available ore ?p) 1)
(increase (resource-use) 2)
))

  ;; C.1: Refining resources.

  (:action burn-coal
   :parameters (?p - place)
   :precondition (and (has-coal-stack ?p)
      (>= (available timber ?p) 1))
   :effect (and (decrease (available timber ?p) 1)
(increase (available coal ?p) 1)
(increase (pollution) 1)))

  (:action saw-wood
   :parameters (?p - place)
   :precondition (and (has-sawmill ?p)
      (>= (available timber ?p) 1))
   :effect (and (decrease (available timber ?p) 1)
(increase (available wood ?p) 1)))

  (:action make-iron
   :parameters (?p - place)
   :precondition (and (has-ironworks ?p)
      (>= (available ore ?p) 1)
      (>= (available coal ?p) 2))
   :effect (and (decrease (available ore ?p) 1)
```

```
(decrease (available coal ?p) 2)
(increase (available iron ?p) 1)
(increase (pollution) 2)))

   )
```

Appendix B

Trader Domain

```
(define (domain batter-trader)
(:requirements :typing :fluents)
(:types village market - location
person carrier - locatable
donkey basket - carrier
goods
)

(:predicates
      (is-at ?p - locatable ?l - location)
(connected-by-trail ?l1 - location ?l2 - location)
(connected-by-stream ?l1 ?l2 - location)
(connected-by-rope-bridge ?l1 ?l2 - location)
(connected-by-wooden-bridge ?l1 ?l2 - location)
(hungry-donkey ?d - donkey ?l - location)
(full-donkey ?d - donkey ?l - location)
(trading ?g1 ?g2 - goods ?m - market)
(buy ?g - goods ?m - market)


)

(:functions
(on-sale ?g - goods ?m - market)
     (sell-price ?g - goods ?m - market)
     (buy-price ?g - goods ?m - market)
```

```
     (available ?g - goods ?l - location)
(quantity-in-carrier ?c - locatable ?g - goods)
(space-in ?c - locatable)
     (cash)
(liquidity)
(expenditure)
(resource)
(labour)
(transport-cost)
)

(:action load-donkey
  :parameters (?d - donkey ?l - location ?g - goods)
:precondition (and
(is-at ?d ?l)
(>(space-in ?d)0)
(>(available ?g ?l)0)
            )
  :effect      (and
(decrease (space-in ?d)1)
(decrease(available ?g ?l)1)
(increase (quantity-in-carrier ?d ?g)1)
(increase (resource)1)
     )
)

(:action load-basket
  :parameters (?b - basket ?p - person ?l - location ?g - goods)
:precondition (and
(is-at ?b ?l)
(>(space-in ?b)0)
(>(available ?g ?l)0)
            )
  :effect      (and
(decrease(space-in ?b)1)
(decrease(available ?g ?l)1)
(increase (quantity-in-carrier ?b ?g)1)
```

```
(increase (labour)1)
     )
)

(:action moving
:parameters (?p - person ?c - carrier ?l1 ?l2 - location)
:precondition (and
(is-at ?p ?l1)
(is-at ?c ?l1)
     )
:effect     (and
(not(is-at ?p ?l1))
(not(is-at ?c ?l1))
(is-at ?p ?l2)
(is-at ?c ?l2)
    )
)

(:action unload
:parameters (?c - carrier ?l - location ?g - goods ?p - person)
:precondition (and
 (is-at ?c ?l) (is-at ?p ?l)
 (>(quantity-in-carrier ?c ?g)0)
      )
:effect     (and
(decrease (quantity-in-carrier ?c ?g)1)
(increase (space-in ?c)1)
(increase (available ?g ?l)1)
    )
)

(:action buy-cash
:parameters (?p - person ?g - goods ?m - market)
  :precondition (and
    (is-at ?p ?m)
    (> (on-sale ?g ?m) 0)
(>=(cash)(sell-price ?g ?m))
```

```
            )
  :effect            (and
    (decrease (on-sale ?g ?m) 1)
          (increase (available ?g ?m)1)
    (decrease (cash)(sell-price ?g ?m))
(increase (expenditure)(sell-price ?g ?m))
    )
)

(:action exchange-goods
:parameters (?p - person ?g1 ?g2 - goods ?m - market)
  :precondition (and
    (is-at ?p ?m)
(trading ?g1 ?g2 ?m)
(>(available  ?g1 ?m )0)
    (> (on-sale ?g2 ?m) 0)
             )
  :effect            (and
    (decrease (on-sale ?g2 ?m) 1)
          (decrease (available ?g1 ?m)1)
(increase (available ?g2 ?m)1)
(increase(liquidity)1)
    )
)

(:action sell-goods-get-cash
  :parameters (?p - person ?g - goods ?m - market)
  :precondition (and
     (is-at ?p ?m)
     (buy ?g ?m)
     (>(available ?g ?m)0)
           )
  :effect      (and
    (increase (on-sale ?g ?m)1)
    (decrease (available ?g ?m)1)
    (increase (cash)(buy-price ?g ?m))
```

)

)

)

Appendix C

Bread Domain

```
(define (domain bread)
(:requirements :typing :fluents)
(:types kitchen
machine
)

(:predicates
      (ready-to-use ?m - machine)
(need-clean ?m - machine)

)

(:functions
(has-flour ?k - kitchen)
(ready-mix ?k - kitchen)
(ready-dough ?k - kitchen)
(loaf-bread ?k - kitchen)
(breakfast-bun ?k - kitchen)
(cooked-bun ?k - kitchen)
(cooked-bread ?k - kitchen)
(labour)
(energy)
(pollution)
)

(:action prepare-mix
```

```
:parameters(?k - kitchen)
:precondition (and
(>=(has-flour ?k)1)
      )
:effect (and
       (increase (ready-mix ?k)1)
       (decrease (has-flour ?k)1)
)
)

(:action kneed-dough-machine
:parameters (?k - kitchen ?m - machine)
:precondition (and
       (>=(ready-mix ?k )2)
(ready-to-use ?m)
      )
:effect (and
(decrease(ready-mix ?k)2)
(increase(ready-dough ?k)2)
(increase(energy)1)
(need-clean ?m)
(not(ready-to-use ?m))
)
)

(:action clean-machine
:parameters (?m - machine)
:precondition (and
(need-clean ?m)
      )
:effect (and
(not(need-clean ?m))
(ready-to-use ?m)
)
)

(:action kneed-hand
```

```
:parameters(?k - kitchen)
:precondition (and
(>=(ready-mix ?k)1)
      )
:effect (and
(decrease(ready-mix ?k)1)
(increase(ready-dough ?k)1)
(increase(labour)1)
)
)

(:action making-loaf-bread
:parameters (?k - kitchen)
:precondition (and
(>=(ready-dough?k)1)
)
:effect (and
(decrease(ready-dough ?k)1)
(increase(loaf-bread ?k)2)
)
)

(:action making-bun
:parameters (?k - kitchen)
:precondition (and
(>=(ready-dough ?k)1)
      )
:effect (and
     (decrease(ready-dough ?k)1)
     (increase(breakfast-bun ?k)5)
)
)

(:action baking-oven-bun
:parameters (?k - kitchen)
:precondition (and
(>=(breakfast-bun ?k)10)
```

```
       )
:effect      (and
(decrease(breakfast-bun ?k)10)
(increase(cooked-bun ?k)10)
(increase(energy)1)
      )
)

(:action baking-oven-loaf-bread
:parameters (?k - kitchen)
:precondition (and
(>=(loaf-bread ?k)4)
       )
:effect      (and
(decrease(loaf-bread ?k)4)
(increase(cooked-bread ?k)4)
(increase(energy)1)
      )
)

(:action baking-charcoal-bread
:parameters (?k - kitchen)
:precondition(and
(>=(loaf-bread ?k)2)
       )
:effect      (and
(decrease(loaf-bread ?k)2)
(increase(cooked-bread ?k)2)
(increase(pollution)1)
      )
)

(:action baking-charcoal-bun
:parameters (?k - kitchen)
:precondition(and
(>=(breakfast-bun ?k)2)
       )
```

```
:effect       (and
(decrease(breakfast-bun ?k)2)
(increase(cooked-bun ?k)2)
(increase(pollution)1)
      )
)
)
```

Appendix D

Production Domain

```
``(define (domain complex-production)
(:requirements :typing :fluents :equality)
(:types
    raw-material refine-material cut-material clean-material part recycle
    assembly-machine process-machine cutting-machine tool part-machine ove
)


(:predicates

(refine-process ?m - machine ?r - refine-material)
(transform-material ?r - raw-material ?rf - refine-material)
(busy ?m - machine)
(recycle-material ?rc - recycle ?rw - raw-material)
(cut-item ?m - material)
(available ?m - machine)
(current-tool ?t - tool)
(item-cut-make ?rf - refine-material ?ct - cut-material)
(cutting-tool-produce ?t - tool ?ct - cut-material)
(made-from ?p - part ?ct - cut-material)
(combine-item ?cp - co-product ?ct - cut-material ?p - part)
(changing-from ?t1 - tool ?t2 - tool)
   (turn-to ?ct - cut-material ?cl - clean-material)
(ready-chemical ?c - item) (prepare-chemical ?c - item)
(produce-to ?cl - clean-material ?p - part)
(warm ?o - oven)
```

```
(cool ?o - oven)

)

(:functions
(instore ?m - material)
(capacity-process ?f - refine-material)
(has-instore ?m - material)
(machine-cost)
(quantity-need ?cy - recycle)
(require-part ?ct - co-product ?p - part)
(changing-tool-cost)
(clean-part)
(product-cost)
(labour)
(hazard) (energy)


)

(:constants product1 product2 uncoated-product2    - product
         metal-part plastic-part coated-plastic      - part
      shaped-plastic shaped-metal   - cut-material
    recycle-plastic recycle-metal - recycle
    substitute   - co-product
    clean-metal clean-plastic   - clean-material
)

(:action setting-machine
:parameters (?m - machine )
:precondition (and
    (busy ?m)
      )
:effect       (and
   (available ?m)
   (not (busy ?m))
)
```

```
)

;; process raw-material

(:action process-raw-material
:parameters (?m - process-machine ?r - raw-material  ?f - refine-material)
:precondition (and  (available ?m)
    (refine-process ?m ?f )
    (transform-material ?r ?f)
    (>=(has-instore ?r)1)
)
:effect       (and  (not (available ?m))
    (busy ?m)
    (increase (instore ?f)(capacity-process ?f))
    (decrease (has-instore ?r) 1)
    (increase(hazard)1)
      )
)

;; process-recycle material

  (:action part-from-recycle
:parameters (?m - process-machine ?cy - recycle ?r - raw-material ?f - ref
:precondition (and  (available ?m)
    (recycle-material ?cy ?r)
    (refine-process ?m ?f )
    (transform-material ?r ?f)
    (>=(instore ?cy)(quantity-need ?cy))
)
:effect       (and  (not (available ?m))
    (busy ?m)
    (increase (instore ?f)1)
    (decrease (instore ?cy)(quantity-need ?cy))
      )
)

;; cutting process
```

```
(:action cutting-process
:parameters (?m - cutting-machine ?r - refine-material ?t - tool ?p - cut-
:precondition (and (available ?m)
   (current-tool ?t)
   (item-cut-make ?r ?p)
   (cutting-tool-produce ?t ?p)
   (>=(instore ?r)1)
     )
:effect     (and   (not (available ?m))
   (busy ?m)
           (increase (instore ?p)2)
           (decrease (instore ?r)1)
     )
)

(:action changing-tool
:parameters ( ?prev-tool - tool  ?next-tool - tool)
:precondition (and
   (current-tool ?prev-tool)
   (changing-from ?prev-tool ?next-tool)
      )
:effect (and
      (current-tool  ?next-tool)
     (not (current-tool ?prev-tool))
)
)

;; making metal-parts

(:action process-shaped-metal
:parameters (?m - part-machine ?cl - clean-material)
:precondition  (and (available ?m)
    (turn-to shaped-metal ?cl)
    (>=(instore shaped-metal)1)
)
:effect       ( and (increase (instore ?cl)2)
```

```
    (decrease (instore shaped-metal)1)
    (busy ?m)
    (not(available ?m))
    (increase(instore recycle-metal)1)
)
)

(:action soak-chemical
:parameters (?cl - clean-material ?c - item)
:precondition (and
   (ready-chemical ?c)
   (produce-to ?cl metal-part)
   (>=(instore ?cl)1)
      )
:effect  (and
  (increase (instore metal-part)2)
          (decrease (instore ?cl)1)
  (not (ready-chemical ?c))
  (prepare-chemical ?c)
  (increase (hazard)1)
 )
)

(:action mix-chemical
:parameters (?c -item)
:precondition (and
(prepare-chemical ?c)
      )
:effect     (and
(ready-chemical ?c)
    )
        )

(:action painting-metal-part
:parameters (?c - coater ?cl - clean-material)
:precondition (and
(available ?c)
```

```
(produce-to ?cl metal-part)
(>=(instore ?cl)1)
      )
:effect
 (and
(decrease(instore ?cl)1)
(increase (instore metal-part)2)
(busy ?c) (not (available ?c))
(increase(labour)1)
 )
)

;; making plastic-part

(:action process-shaped-plastic
:parameters (?m - part-machine ?cl - clean-material)
:precondition  (and
    (available ?m)
    (turn-to shaped-plastic ?cl)
    (>=(instore shaped-plastic)1)
)
:effect ( and
(increase (instore ?cl)4)
(decrease (instore shaped-plastic)1)
(busy ?m)
(not (available ?m))
(increase (instore recycle-plastic)1)
)
)

(:action coating-plastic-part
:parameters (?c - coater ?cl - clean-material)
:precondition (and
(available ?c)
(produce-to ?cl plastic-part)
(>=(instore ?cl)1)
      )
```

```
:effect
 (and
(decrease(instore ?cl)1)
(increase (instore plastic-part)2)
(increase (instore recycle-plastic)1)
(busy ?c) (not (available ?c))
 )
)

;; assembly product


(:action warming-oven
:parameters (?o - oven )
:precondition (and
 (cool ?o)
      )
:effect       (and
(warm ?o)
      )
)

(:action assemble-product1-formula1
    :parameters ( ?m - assembly-machine ?o - oven)
    :precondition (and (available ?m)
   (>= (instore plastic-part)2)
   (>= (instore metal-part)2)
   (warm ?o)
       )

   :effect       (and (increase (instore product1)2)
         (decrease (instore plastic-part)2)
   (decrease (instore metal-part)2)
   (not(available ?m))
   (busy ?m)
   (not(warm ?o))
   (cool ?o)
```

```
   (increase (machine-cost)10)
      )
)


(:action assemble-product1-formula2
:parameters ()
:precondition (and
   (>=(instore metal-part)1)
   (>=(instore clean-plastic)1)
      )
:effect     (and  (increase (instore product1)1)
  (decrease (instore metal-part)1)
  (decrease (instore clean-plastic)1)
  (increase (labour)1)
    )
)

(:action assemble-product1-formula3
:parameters (?it - item)
:precondition (and
   (>=(instore recycle-metal)1)
   (>=(instore clean-plastic)1)
   (ready-chemical ?it)
      )
:effect     (and  (increase (instore product1)1)
  (decrease (instore metal-part)1)
  (decrease (instore clean-plastic)1)
  (increase (hazard)1)
    )
)

(:action assemble-product2-formula1
    :parameters (?m - assembly-machine)
    :precondition (and (available ?m)
   (>=(instore plastic-part)2)
   (>=(instore clean-plastic)1)
```

```
)
    :effect      (and (increase (instore product2)3)
  (decrease (instore plastic-part)2)
  (decrease (instore clean-plastic)1)
  (not (available ?m))
  (busy ?m)
  (increase(machine-cost)10)

    )
)

(:action assemble-product2-formula2
:parameters ()
:precondition (and
     (>=(instore plastic-part)4)
     (>=(instore recycle-plastic)2)
      )
:effect (and (decrease (instore plastic-part)4)
     (increase (instore uncoated-product2)2)
     (decrease(instore recycle-plastic)2)
     (increase (labour)1)
)
)

(:action coating-product2
:parameters (?c - coater)
:precondition (and
 (available ?c)
 (>(instore uncoated-product2)2)
)
:effect       (and
  (decrease (instore uncoated-product2)2)
  (increase (instore product2)2)
  (busy ?c)
  (not (available ?c))
      )
```

)

)

Appendix E

Sugar Domain

```
(define (domain supply-chain)
(:requirements :typing :fluents :equality)
(:types
 brand raw-cane        - sugar
 mill depot          - location
 truck crane         - loader
farm field
)

(:predicates
(available ?m - mill)
(has-resource ?r - raw-cane ?m - mill)
(produce ?m - mill ?b - brand)
(current-process ?m - mill ?b - brand)
(change-process ?b1 ?b2 - brand)
(place-order ?r - raw-cane ?m - mill)
(transport-to ?r - raw-cane  ?m - mill)
(at-location ?d - loader  ?l - location)
(connected ?l1 ?l2 - location)
(busy ?m - mill)
(ready-crane ?c - crane)
(service-crane ?c - crane)
)

(:functions
(mill-cost) (cost-process ?m - mill)
```

```
(process-cost ?m - mill)
(resource-use)
(unharvest-field)
(truck-cap ?t - truck)
(in-truck-sugar ?b - brand ?t - truck)
(in-storage ?m - location ?b - brand)
(total-distance)
(distance ?l1 ?l2 - location)
(has-resource ?r - raw-cane ?m - mill)
(max-changing ?m - mill)
(inventory-cost)
(changing-product)
(capacity ?c - crane)
(max-service-time ?c - crane)
(service-time ?c - crane)
(handling-cost)
(max-produce ?m - mill)
(labour-cost)


)


(:action produce_sugar
:parameters (?r - raw-cane ?m - mill ?b - brand)
:precondition (and
(current-process ?m ?b)
(available ?m)
(produce ?m ?b)
(>(has-resource ?r ?m)0)
(>(max-changing ?m)0)
          )
:effect      (and
(increase (in-storage ?m ?b)1)
(decrease (has-resource ?r ?m)1)
(busy ?m)
(not(available ?m))
```

```
(increase (mill-cost)(cost-process ?m))
          )
)


(:action produce_sugar_max
:parameters (?r - raw-cane ?m - mill ?b - brand)
:precondition (and
(current-process ?m ?b)
(available ?m)
(produce ?m ?b)
        (>=(has-resource ?r ?m)(max-produce ?m))
(>(max-changing ?m)0)
          )
:effect      (and
(increase (in-storage ?m ?b)(max-produce ?m))
(decrease (has-resource ?r ?m)(max-produce ?m))
(busy ?m)
(not(available ?m))
(increase (mill-cost)(*5(cost-process ?m)))
          )
)

(:action order-sugar-cane
:parameters (?r - raw-cane ?m - mill )
:precondition (and
(>=(has-resource ?r ?m)0)
(<=(has-resource ?r ?m)0)
      )
:effect       (and
 (place-order ?r ?m)
      )
)

(:action setting-machine
:parameters (?m - mill)
:precondition (and
```

```
 (busy ?m)
      )
:effect       (and
 (not (busy ?m))
 (available ?m)
      )
)

(:action change-product
:parameters (?m - mill ?b1 - brand ?b2 - brand)
:precondition (and
 (current-process ?m ?b1)
 (change-process ?b1 ?b2)
      )
:effect       (and
(current-process ?m ?b2)
(not(current-process ?m ?b1))
(decrease(max-changing ?m)1)
      )
)

; sugar cane can be obtained from the farm or from other mills

(:action sugar-cane-farm
:parameters (?r - raw-cane ?m - mill)
:precondition (and
(place-order ?r ?m)
(>(unharvest-field)0)
      )
:effect       (and
(decrease (unharvest-field)1)
(increase (has-resource ?r ?m)5)
(not (place-order ?r ?m))
(increase (inventory-cost)10)
      )
)
```

```
(:action sugar-cane-mills
:parameters (?r - raw-cane ?m1 ?m2 - mill)
:precondition (and
(place-order ?r ?m1)
(>(has-resource ?r ?m2)0)
      )
:effect       (and
(increase (has-resource ?r ?m1)1)
(decrease (has-resource ?r ?m2)1)
(not (place-order ?r ?m1))
(decrease(inventory-cost)1)
      )
)

(:action load_truck_crane
:parameters (?b - brand ?t - truck ?l - location ?c - crane)
:precondition (and
 (at-location ?t ?l)
 (at-location ?c ?l)
 (>=(in-storage ?l ?b)(capacity ?c))
 (>=(truck-cap ?t)(capacity ?c))
 (ready-crane ?c)

      )
:effect      (and
 (decrease (in-storage ?l ?b)(capacity ?c))
 (decrease (truck-cap ?t)(capacity ?c))
 (increase (in-truck-sugar ?b ?t)(capacity ?c))
 (increase (handling-cost)10)

    )
)

(:action check-service
:parameters (?c - crane ?l - location)
:precondition (and
```

```
 (at-location ?c ?l)
 (>=(service-time ?c)0)
 (<=(service-time ?c)0)
      )
:effect       (and
 (not(ready-crane ?c))
     (service-crane ?c)
     (increase(service-time ?c)(max-service-time ?c))
      )
)

(:action maintainence-crane
:parameters (?c - crane ?l - location)
:precondition (and
 (at-location ?c ?l)
 (service-crane ?c)
      )
:effect      (and
(ready-crane ?c)
     )
)

(:action load-truck-manual
:parameters (?b - brand ?t - truck ?l - location)
:precondition (and
 (at-location ?t ?l)
 (>(in-storage ?l ?b)0)
 (>(truck-cap ?t)0)

      )
:effect      (and
 (decrease (in-storage ?l ?b)1)
 (decrease (truck-cap ?t)1)
 (increase (in-truck-sugar ?b ?t)1)
 (increase (handling-cost)1)
    )
)
```

```
(:action drive_truck
:parameters (?t - truck ?y1 ?y2 - location)
:precondition (and
     (at-location  ?t ?y1)
     (connected ?y1 ?y2)
)
:effect  (and (at-location ?t ?y2)
     (not(at-location ?t ?y1))
)
)

(:action unload_truck
:parameters (?b - brand ?t - truck ?l - location)
:precondition (and
(at-location ?t ?l)
(>(in-truck-sugar ?b ?t)0)
      )
:effect (and
       (increase (in-storage ?l ?b)1)
(decrease (in-truck-sugar ?b ?t)1)
(increase (truck-cap ?t)1)
)
)

)
```

Bibliography

[1] D. L. A. S. Alfonso Gerevini, Patrik Haslum and Y. Dimopoulos. Deterministic Planning in the Fifth International Planning Competition: PDDL3 and Experimental Evaluation of the Planners. *Artificial Inteligence*, 173:pages 619–668, 2009.

[2] F. Bacchus. AIPS 2000 Planning Competition: The Fifth International Conference on Artificial Intelligence Planning and Scheduling Systems. *AI Magazine*, 22:47–55, 2001.

[3] J. Benton, M. van den Briel, and S. Kambhampati. A Hybrid Linear Programming and Relaxed Plan Heuristic for Partial Satisfaction Planning Problems. In *Proceedings of ICAPS*, pages 34–41, 2007.

[4] A. L. Blum and M. L. Furst. Fast Planning through Planning Graph Analysis. *Artificial Intelligence*, 90:1636–1642, 1997.

[5] B. Bonet. A Robust and Fast Action Selection Mechanism for Planning. In *Proceedings of AAAI-97*, pages 714–719. MIT Press, 1997.

[6] B. Bonnet and H. Geffner. HSP: Heuristic Search Planner. In *AIPS-98 Planning Competition*, 1998.

[7] T. Bylander. The Computational Complexity of Propositional STRIPS Planning. *Artificial Intelligence*, 69(1-2):165 – 204, 1994.

[8] Chih-Wei, B. Wah, Ruoyun, and Y. Chen. Handling Soft Constraints and Goal Preferences in SGPLAN. In *ICAPS Workshop on Preferences and Soft Constraints in Planning*. ICAP, 2006.

[9] C. A. C. Coello and L. Nacional. An Updated Survey of GA-Based Multiobjective Optimization Techniques. *ACM Computing Surveys*, 32:109–143, 1998.

[10] A. I. Coles, M. Fox, D. Long, and A. J. Smith. A Hybrid Relaxed Planning Graph-LP Heuristic for Numeric Planning Domains. In *Proceedings of the Eighteenth*

International Conference on Automated Planning and Scheduling (ICAPS 08), September 2008.

[11] T. Dean and S. Kambhampati. Planning and Scheduling. In *CRC Handbook of Computer Science and Engineering*. CRC Press, 1996.

[12] Y. Dimopoulos, A. Gerevini, P. Haslum, and A. Saetti. The Benchmark Domains of the Deterministic Part of IPC-5. In *Booklet of the 2006 Planning Competition, ICAPS'06*, 2006.

[13] B. Drabble, A. Tate, and S. Bridge. The Use of Optimistic and Pessimistic Resource Profiles to Inform Search in an Activity Based Planner. In *Proceedings of AIPS-94*, pages 243–248. AAAI Press, 1994.

[14] S. Edelkamp. Mixed Propositional and Numerical Planning in the Model Checking Integrated Planning System. In *AIPS-2002 Workshop on Temporal Planning*, 2001.

[15] S. Edelkamp. Planning with Pattern Databases. In *Proceedings Of The 6th European Conference On Planning (ECP-01)*, pages 13–24, 2001.

[16] S. Edelkamp. Promela Planning. In *Proceedings of SPIN-03*, pages 197–212, 2003.

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

[18] S. Edelkamp and S. Jabbar. MIPS-XXL: Featuring External Shortest Path Search for Sequential Optimal Plans and External Branch-And-Bound for Optimal Net Benefit. In *6th International Planning Competition Booklet (ICAPS-08)*, 2008.

[19] M. Ehrgott. A Discussion of Scalarization Techniques for Multiple Objective Integer Programming. *Annals of Operations Research*, 147:343–360, October 2006.

[20] G. W. Evans. An Overview of Techniques for Solving Multiobjective Mathematical Programs. *Management Science*, 30(11):1268–1282, 1984.

[21] R. E. Fikes and N. J. Nilsson. STRIPS: A New Approach to the Application of Theorem Proving to Problem Solving. *Artificial Intelligenc*, 2, 1971.

[22] C. M. Fonseca and P. J. Fleming. Genetic Algorithms for Multiobjective Optimization: Formulation Discussion and Generalization. In *Proceedings of the 5th International Conference on Genetic Algorithms*, pages 416–423, San Francisco, CA, USA, 1993. Morgan Kaufmann Publishers Inc.

[23] M. Fox and D. Long. The Automatic Inference of State Invariants in TIM. *Journal of Artificial Intelligence Research*, 9:367–421, 1998.

[24] M. Fox and D. Long. PDDL2.1: An Extension of PDDL for Expressing Temporal Planning Domain. *Journal of Artificial Intelligence Research*, 20:61–124, 2003.

[25] M. S. Fox and N. Sadeh-Koniecpol. Why is Scheduling So difficult? a CSP Perspective. In *Proceedings of the European Conference On Artificial Intelligence*, pages 754 – 767, 1990.

[26] M. R. Garey, D. S. Johnson, and R. Sethi. The complexity of flowshop and jobshop scheduling. *Mathematics of Operations Research*, 1(2):117–129, 1976.

[27] A. Garrido and F. Barber. Integrating Planning and Scheduling. *Applied Artificial Intelligence*, 15:471–491, 2001.

[28] A. Garrido and D. Long. Planning with Numeric Variables in Multiobjective Planning. In *16th European Conference on Artificial Intelligence (ECAI -2002)*, pages 662–666, 2004.

[29] A. Garrido and E. Onaindia. On the Application of Least-commitment and Heuristic Search in Temporal Planning. In *IJCAI-2003*, 2003.

[30] A. Garrido, M. Salido, F. Barber, and M. López. Heuristic Methods for Solving Job-Shop Scheduling Problems, 2000.

[31] A. Gerevini and D. Long. Plan Constraints and Preferences in PDDL3. Technical report, Department of Electronics for Automation, University of Brescia, Italy, 2005.

[32] A. Gerevini, A. Saetti, and I. Serina. An Approach to Temporal Planning and Scheduling in Domains with Predictable Exogenous Events. *Journal of Artificial Intelligence Research*, 25:187–231, 2004.

[33] A. Gerevini, A. Saetti, and I. Serina. Planning with Numerical Expressions in LPG. In *Proceedings of the 16th European Conference on Artificial Intelligence (ECAI-04)*. IOS-Press, Valencia, Spain, 2004.

[34] M. Ghallab, A. Howe, C. Knoblock, D. Mcdermott, M. V. Ashwin Ram, D. Weld, and D. Wilkins. PDDL— The Planning Domain Definition Language. Technical report, Yale Center for Computational Vision and Control, 1998.

[35] M. Ghallab, D. Nau, and P. Traverso. *Automated Planning Theory and Practice*. Morgan Kaufman, 2004.

[36] J. Hoffman, S. Edelkamp, S. Thiebaux, F. S. Liporace, and S. Trug. Engineering Benchmarks for Planning: The domains used in the Deterministic part of IPC-4. *Journal of Artificial Intelegence Research*, 26:453–541, 2006.

[37] J. Hoffmann. Extending FF to Numerical State Variables. In *Proceedings of the 15th European Conference on Artificial Intelligence*, pages 571–575. Wiley, 2002.

[38] J. Hoffmann. The Metric-FF Planning System: Translating Ignoring Delete List to Numeric State Variable. *Journal of Artificial Intelligence Research*, 20:291–341, 2003.

[39] J. Hoffmann and S. Edelkamp. The Deterministic Part of IPC-4: An Overview. *Journal of Artificial Inteligence Research*, 24:pages 519–579, 2005.

[40] J. Hoffmann and B. Nebel. The FF Planning System: Fast Plan Generation Through Heuristic Search. *Journal of Artificial Intelligence Research*, 14:253–302, 2001.

[41] R. Howey, D. Long, and M. Fox. VAL: Automatic Plan Validation, Continuous Effects and Mixed Initiative Planning Using PDDL. In *ICTAI '04: Proceedings of the 16th IEEE International Conference on Tools with Artificial Intelligence*, pages 294–301, Washington, DC, USA, 2004. IEEE Computer Society.

[42] H. Kautz and J. P. Walser. Integer Optimization Models of AI Planning Problems. *The Knowledge Engineering Review*, 15:2000, 2000.

[43] J. Koehler. Planning under Resource Constraints. In *Proceedings of the 15th European Conference on AI*, 1998.

[44] P. Laborie and M. Ghallab. Planning with Sharable Resource Constraints. In *IJCAI*, pages 1643–1651, 1995.

[45] E. R. Lieberman. Soviet Multi-Objective Mathematical Programming Methods: An Overview. *Management Science*, 37(9):1147–1165, 1991.

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

[47] D. Long, M. Fox, L. Sebastia, and A. Coddington. An Examination of Resources in Planning. In *In Proc. of 19th UK Planning and Scheduling Workshop, Milton Keynes*, 2000.

[48] D. Mcdermott. A Heuristic Estimator for Means-Ends Analysis in Planning. In *Proceeding Third International Conference on AI Planning Systems AIPS-96*, pages 142–149, 1996.

[49] D. Mcdermott. The 1998 AI Planning Systems Competition. *AI Magazine*, 21:35–55, 2000.

[50] A. Osyczka. An Approach to Multicriterion Optimization Problems for Engineering Design. *Computer Methods in Applied Mechanics and Engineering*, 15(3):309 – 333, 1978.

[51] E. P. Pednault. ADL:Exploring the Middle Ground between STRIPS and the Situation Calculus. In *Proceedings of the first International Conference on Principles of knowledge Representation and Reasoning*, pages 324–332, San Francisco, CA, USA, 1989. Morgan Kaufmann Publisher Inc.

[52] J. S. Penberthy and D. S. Weld. Temporal Planning with Constraints (preliminary report), 1993.

[53] I. Refanidis and I. Vlahavas. GRT: A Domain Independent Heuristic for STRIPS Worlds based on Greedy Regression Tables. In *Proceedings of European Conference On Planning ECP-99*, pages 346–358. Springer, 1999.

[54] I. Refanidis and I. Vlahavas. Multiobjective Heuristic State-Space Planning. *Artificial Intelligence*, 145:1–32, 2003.

[55] J.-A. Shin and E. Davis. Processes and Continuous Change in a SAT-based planner. *Artificial Intelligence*, 166(1-2):194 – 253, 2005.

[56] D. E. Smith, J. Frank, and A. K. Jnsson. Bridging the Gap Between Planning and Scheduling. *The Knowledge Engineering Review*, 15:47–83, 2000.

[57] D. E. Smith, J. Frank, and A. K. Jönsson. Bridging the Gap between Planning and Scheduling. *Knowledge Engineering Review*, 15:2000, 2000.

[58] B. Srivastava and S. Kambhampati. Efficient Planning through Separate Resource Scheduling. In *Proceedings of the AAAI Spring Symp. on.* AAAI Press, 1999.

[59] B. Srivastava and S. Kambhampati. Scaling up Planning by Teasing out Resource Scheduling. *Lecture Notes in Computer Science*, 1809/2000:172–186, 2000.

[60] M. van den Briel, J. Benton, S. Kambhampati, and T. Vossen. An LP-based Heuristic for Optimal Planning. *Lecture Notes in Computer Science*, 4741/2007:651–665, 2007.

[61] D. S. Weld. An Introduction to Least Commitment Planning. *AI Magazine*, 4, 1994.

[62] S. A. Wolfman and D. S. Weld. Combining Linear Programming and Satisfiability Solving for Resource Planning. *The Knowledge Engineering Review*, 15:2000, 2000.

[63] S. A. Wolfman and D. S. Weld. Combining Linear Programming and Satisfiability Solving for Resource Planning. *The Knowledge Engineering Review*, 16:85–99, 2001.

[64] H. Yan, Q. Wei, and J. Wang. Constructing Efficient Solutions Structure of Multiobjective Linear Programming. *Journal of Mathematical Analysis and Applications*, 307(2):504 – 523, 2005.