Multi-objective planning using a metric sensitive planner

Sroka, Michal

Awarding institution:
King's College London

Multi-objective planning using a metric sensitive planner.

by

Michal Sroka

A thesis submitted in partial fulfillment for the degree of Doctor of Philosophy

in the

Department of Informatics

in the

School of Natural & Mathematical Sciences

October 2014

Abstract

Automated planning addresses the problem of generating a sequence of actions to satisfy given goal conditions for a constructed model of the world. In recent planning approaches heuristic guidance is used to lead the search towards the goal. The focus of this work is on domains where plan quality is assessed with plan metrics. A discussion of the impact of a popular relaxed planning graph heuristic on the quality of plans in such domains is presented. The relaxed planning graph heuristic bias towards shorter plans, irrespective of quality, is described. A novel approach to constructing the relaxed planning graph based on metric cost is presented to overcome this bias and to generate good quality plans. A notion of metric sensitivity as the ability of a planner to respond to the change of the plan metric, is introduced and methods to determine metric sensitivity are presented. Current state-of-the-art planners are evaluated in terms of their metric sensitivity. This research also tackles the problem of planning in multi-objective domains, where quality of a plan is evaluated using multiple plan metrics. For multi-objective domains the solution is no longer a single plan but a set of plans. A set of non dominated solutions is called a pareto frontier. This thesis contains a discussion on the desired properties of such sets of plans and methods of generating them. Metric sensitivity is a required property for a planner to effectively reason with user defined metrics and generate desired set of plans. The main significant contributions of the work described in the thesis are:

1. A definition and exploration of metric sensitivity in planning.
2. A context-dependent, cost-based relaxed planning graph and heuristic.
3. A compilation method from cost to temporal domains.
4. Examination of the impact of planners' properties on the quality of plans and APFs.

Acknowledgements

I would like to use this opportunity to thank my doctoral supervisors: Derek Long, Maria Fox and Kerem Akartunali. Thank you for your support, encouragement, guidance and your patience. You have been always available and happy to help. I would like to extend a special gratitude to Derek for many productive meetings, fruitful discussions and seeding the key ideas which shaped this research. Thank you for your time, support, openness, sharing expertise and ideas.

I am especially grateful to have received a tremendous amounts of support from Margot and my family. Thank you for your support, being here when I needed, helping me out with a good word and many suggestions. You have inspired, motivated and encouraged me throughout the time of this research. Only with your support I could be persistent and arrive at this stage.

And finally, I would like to thank my friends who have helped me through this research. Special thanks to the current and former students of the informatics department at King's College and Strathclyde University. Thank you for inspirational discussions, presentations and keeping the amazing atmosphere at the department. Widening my horizons with you, either at the department or conferences, was a great pleasure.

Contents

List of Figures

List of Tables

Glossary

The following symbols are used throughout this thesis:

- π - denotes a plan
- Π - set of plans.
- ϕ - A set of plans containing only non-dominated solutions, also called an Approximation of Pareto Frontier (APF).
- Θ - denote a cost under specific metric function, $\Theta_i = cost(\pi, m_i)$ is the value of the ith metric for a plan π. Θ is used interchangeably with m, $\Theta(\pi) = m(\pi)$.
- I^{name} - indicator for evaluation of frontiers of plans.
- $d^{name}(\pi_1, \pi_2)$ - distance measure between two plans.
- $h_{name}(s)$ - is a heuristic function used to evaluate states.
- PF - Pareto Frontier.
- N - usually denotes number of metric functions.
- α - is used to denote weights on metric functions.
- α_i - is the weight on i-th function.
- Metric fluent - is a metric function used in planning. Metric fluents are defined in PDDL 2.1.
- Plan metric - is a function for evaluation of plans which is a combination of metric fluents into a single metric.

Chapter 1

Introduction

1.1 Automated planning

Automated planning is the process of constructing a structured collection of actions which, when applied in order, will transform a given initial situation to a state satisfying a specified goal. The task of finding plans is carried out by a program called a planner. An input for the planner is a domain and problem description. The aim of the planner is to produce a plan that satisfies the goal condition. A domain description consists of actions, object types, predicates, constants and functions. A problem description consists of an initial state and a goal, where the goal is a partial state. The problem description can also contain preferences on the plan trajectory expressed as propositions desired to be true or desired values of functions.

Domain independent planning is ultimately concerned with modelling and solving real world problems. These problems usually have a rich structure and context which cannot be fully modelled. Some level of abstraction is required. Depending on the intent of the user, different degrees of abstraction are appropriate. A lot of research effort in planning is directed towards the extension of the expressiveness of the current tools and modelling languages.

Numerical functions allow the user to specify interesting numerical aspects of the task which they are interested in solving. They can be combined into a plan metric. Minimisation (or maximisation) of the plan metric is a form of expressing users preferences.

Many current planners, as discussed in Section 3.4, do not consider the plan metrics; although they allow the user to specify the plan metric, it is not a strong driver when selecting actions for the plan. Instead, the approaches used by these planners rely on the correlation between the plan length and the plan cost. This means that they only minimize the plan length as a proxy for minimizing the metric.

In many realistic domains, the action cost depends on the context in which the action is applied. Current cost-optimal planners require the cost of an action to be specified before the planning process starts. A simple example can demonstrate that some action costs cannot be pre-defined. These actions require a context-dependent function to express their cost. Consider a transportation domain with a truck and drive-truck action. The cost of this action depends on the amount of fuel used. Fuel used depends on the distance driven, the load of the truck, and the fuel consumption. The cost of this action cannot be calculated unless all of these components are known. Some of the components, such as the distance between cities and fuel consumption, are known before the planning commences. However, the planner only knows the load of the truck when it knows the state from which it is applied. Therefore the cost of this action cannot be computed prior to the planning stage.

It is often the case that a domain contains multiple conflicting plan metrics. When a user expresses a preference to minimize two or more of these plan metrics, a multi-objective planning problem is formed. A planner solving the multi-objective planning problem should provide its user with the flexibility to specify, as part of the problem description, the set of plan metrics which should be minimized or maximized.

Examination of the current benchmark domains reveals that there are no domains which offer an interesting trade-off at arriving to the goal using distinct plan trajectories. In these domains,

it is often the case that if a domain contains multiple metrics there is, nevertheless, a single solution which dominates all others. The lack of multiple distinct paths is related to the correlation between the cost and the plan length in the current benchmark domains. This means that minimizing the plan length optimizes plan metrics. A truly multi-objective domain provides alternative paths to the goal where at least some paths use different resources. This gives an opportunity to the planner to explore these different solutions.

When optimizing multiple metrics, the solution is no longer a single plan but a set of plans. This set of plans should demonstrate the trade-off between different plan metrics. The aim of a multi-objective planning system is to generate a high quality set of plans which is well distributed across the solution space and offers a good insight into the possible variety of solutions.

1.2 Problem statement

In order to expand the class of effectively solvable numerical planning problems, a novel approach is necessary. The new approach must be sensitive to the metric expressed by the user. It should not depend on the correlation between plan metric and plan length as a way of obtaining high quality plans. Current state-of-the-art and the new proposed solution should be evaluated using an appropriate metric. In order to assess the impact on generating good quality plans in response to the metric function, a study of metric sensitivity is required. The property of metric sensitivity should be defined, along with a method for quantitatively assessing metric sensitivity.

A method should be developed to generate a frontier of plans for a multi-objective planning problem. One possible approach is to use the metric sensitive planner to find good quality plans representing trade-offs between the user defined metrics. Plans from the set should represent various areas of the solution space. There are many multi-objective search algorithms. However, to the best of our knowledge, none of them has been successfully used in planning.

1.3 The aim of this research

One of the main aims of this research is to design and develop an automatic domain independent multi-objective planning system (MOPS). This system provides an insight into the solution space of a problem from the perspective which matter to the user. These perspectives are expressed in plan metric functions. Since a solution to a multi-objective problem is a set of plans, the system produces such sets. A method to evaluate the frontier of plans, based on multiple plan metrics provided, is developed. The system is then evaluated to demonstrate that it can provide a high quality frontier of plans meeting the users' criteria.

In order to reason effectively with multiple metrics, a planner must be metric sensitive. This allows the planner to generate plans in different areas of the search space. Therefore a definition and study of metric sensitivity is required. A novel approach to reasoning with metrics and its evaluation is also presented.

1.3.1 Thesis statement

Metric sensitive planners, in combination with aggregation methods for multiple metrics, are an effective tool for the generation of approximations of pareto frontiers of plans.

1.4 Methodology

Algorithms presented in this thesis are designed to generate good quality solutions, and good quality solution sets. A comparative, empirical study of the quality of the solutions generated is presented in the result section of each chapter. The new approach to producing good quality plans in response to the plan metric is evaluated in comparison with the closest state-of-the-art planner capable of solving similar class of problems. The quality of the plans is assessed using a standard quality measure.

For the purpose of the comparison, as part of this research, a system for running the planners is developed. For each experiment the system runs planners and, according to the experiment strategy selected, compares their output using the specified plan metrics. The quality of plans is assessed solely based on the plan metrics.

The multi-objective planning system developed for the purpose of this thesis is also used to evaluate the quality of frontiers of plans generated by different planners. A survey of quality measures for multi-objective problems is presented. A selection of metrics is used for the comparison.

1.5 Thesis structure

The remaining part of the thesis is structured as follows:

First, in the background chapter, a brief introduction to the field of automated planning is given. This includes a short summary of different versions of the language used for modelling of problems and domains, ways of handling numerical functions and a survey of current domain independent, satisficing planners. A brief history of evolution of the numerical functions within the modelling language is given. This chapter ends with a discussion of multi-objective domains.

A definition of metric sensitivity is given in the third chapter. A discussion of what metric sensitivity is and how it should be measured is also presented. The current state-of-the-art planners are surveyed together with their metric sensitivity evaluation. The chapter also contains a discussion of to what extent the current planners are metric sensitive. A method for obtaining different plans from metric *insensitive* planners, using lower bounds on resources, is given. The best planner is selected for comparison with the new approach. A study of the impact of stochasticity on the planners ability to exhibit a metric sensitive behaviour is examined.

This is followed by a description of a novel cost-based heuristic function. The strengths and weaknesses of this novel approach are discussed and an algorithm for calculating the heuristic is given. The algorithm is presented together with a method of dealing with context-dependent action cost. An implementation of the cost-based heuristic is proposed based on a novel compilation from cost to temporal domains. This compilation is further discussed. Results for the most appropriate state-of-the-art planner and the new approach are then presented. Results for the impact of specific properties of planners, like stochasticity, are also presented.

The fifth chapter contains material related to multi-objective planning. The chapter starts with a background information relevant to multi-objective planning. First, a survey of current methods of generating sets of plans is presented. Methods in this section focus on various aspects of the sets of plans, such as variety or quality. Following that, the section contains a short description of methods used for visualising the frontiers of plans to the user. The presentation of plans is an important way of communicating the trade-offs between plan metrics to the user. Then a survey of methods for evaluation of frontiers of plans is shown. The survey contains different quality indicators used for evaluating different aspects of frontiers of plans.

After the background, we show methods for generating frontiers of plans. After a general description of methods of generating diverse frontiers of plans using weights on plan metrics, we focus on two properties of planners and their impact on the process of generation of frontiers of plans. These two properties are metric sensitivity and stochasticity.

Following this, a multi-objective planning system (MOPS) is described. The system can be configured with any planner and strategy for using the planner. The notion of strategy is explained in the chapter. MOPS is used as a test environment.

The results focus on the difference between properties of frontiers generated by different configurations of MOPS. Metrics used to evaluate these differences are based on the quality indicators described in this thesis. Apart from the examination of different frontier generators, the result section contains comparative study of the impact of particular properties of planners on

the quality of the frontier generated. This includes stochasticity as a means for obtaining high diversity in the solution set. Stochasticity also helps in improving the quality of the frontier. A methods to exploit stochasticity for further improvement of the quality of the frontier is given. Second property examined is metric sensitivity. Based on its definition we expect to see an improvement of the quality of the frontiers. We also compare the impact on results coming from stochasticity with the impact coming from metric sensitivity. The experiment section ends with evaluating how a method of removing under populated methods in the solution space compares to the methods of populating the entire frontier.

Chapter 2

Background

This chapter presents an overview of the current state-of-the-art planning and its main variants including temporal, numeric and deterministic. The section is split into three parts: modelling of planning problems, finding a plan, and result representation. Where the modelling part starts with description of the modelling language used in planning. The extensions, including numerical functions, to the modelling language are presented. Examples of multi-objective domains are given. Following that, in the part concerned with finding the plan, a description of techniques and data structures used in planning is presented. Planners relevant to the research are discussed together with a brief description on how they handle numerical variables. These domains contain interesting trade-offs between different paths of arriving to the goal. This section is concluded with a discussion of the notion of pareto frontier, which is the expected output of a multi-objective planning system. Further background information, relevant to specific chapters, is provided in these chapters.

2.1 Planning

For a given domain and a problem description, containing a start state and a goal description, planning is a process of selecting and ordering a sequence of actions which, when applied in succession, takes us from the start state to the state that satisfies the goal condition. A planning domain contains a high level model of the world including: type hierarchies for objects, actions, predicates constants and functions. It is a lifted representation, meaning that it uses parametrised predicates, fluents and actions. The predicates describe possible facts such as (at ?t - truck ?l - location). This is an example of a parametrised predicate which is used to denote that a particular truck is at a particular location in the problem description, for example: (at truck1 parking). This example predicate represents a situation in which a truck labelled *truck*1 is stationary at a location labelled *parking*. Lifted actions describe families of possible transitions between states. A problem description contains: lists of available objects, lists of grounded predicates which are true in the start state, list of metric fluents with their values and the goal description.

Formally, a planning problem is a tuple <A, P, i, g, Cost(s_i, a_j)> where A is a set of grounded actions, P is a set of propositions, i and $g \subseteq P$ describe the initial situation and goal respectively. A state $s \subset P$ such that all propositions within the subset are considered true and all propositions which do not appear in the subset are considered false. Cost(s_i, a_j), for a given state $s_i \subset P$, and a grounded action $a_j \subset A$, returns the cost of applying the action in the given state.

Applicable(s_i) and Cost(s_i, a_j) are two functions used to determine applicable actions and the cost of applying them, and eventually the cost of the plan. In the simplest case, Cost(s_i, a_j) is constant, or dependent only on a_j.

Each action has an explicitly defined set of preconditions, denoted as $Precondition(a)$, such that the action is only applicable in a state s_i if all of its preconditions are satisfied in the state s_i. After applying an action with positive effects Add(a_j) and delete effects Del(a_j), the new state

is described as $s'_i = s_i \setminus Del(a_j) \cup Add(a_j)$. This operation is called the state transition and is denoted as Transition(s_i, a_j). Transition(s_i, a_j), is a function which for a given state and action to be applied returns a new state which is the result of applying the given action in the given state. Another useful function that can be derived from the set of actions is Applicable(s_i). For a given state $s_i \subset P$, it returns the collection of applicable actions.

In numeric planning the model is extended with a set of numerical functions called fluents V, <V, A, P, i, g, Cost(s_i, a_j)>. Fluents are functions which represent a numerical value for a given state and objects. Objects are conditional parameters of metric fluents. Application of actions can change the values of fluents. Metric fluents can be combined together into a plan metric which is then used to evaluate the quality of the plan.

Definition 2.1. A metric fluent is a numeric state variable.

Definition 2.2. A plan metric is an arithmetic combination of metric fluents and constants. It is used to evaluate the quality of the plan for a given problem instance.

The plan metric, also called a cost function, in general depends on the values of the metric fluents and the state in which it is applied. In practice, simple and state independent cost models are generally used. In classical planning, the quality is usually assessed by the length of the plan, so that $\forall_{i,j} Cost(s_i, a_j) = 1$. For domains where cost is constant, minimizing the plan length also minimizes the cost. The correlation between plan cost and plan length occurs in most of the current benchmark domains. Currently many planners parse specification of action costs using metric fluents. However, context-dependent action cost is ignored. Therefore the cost model which current planners solve effectively does not include state dependent action cost. The cost function satisfies: $\forall_{i,j,k} Cost(s_i, a_j) = Cost(s_k, a_j)$. The interesting problems arise when action cost cannot be computed *apriori*, and differs depending on the assignment of metric fluent values in a state. This research aims at solving problems with context-dependent action costs.

Another extension to the modelling language is to support temporal planning. Temporal planning is designed to handle domains where the duration of actions or their placement in time is important. This can arise when dealing with temporal deadlines or actions which can only be executed during the time when another action is executing (parallel execution). Temporal planning allows specification of propositions which become true or false at a specified point in time. Similarly, a metric fluents can be assigned a specific value at a given time point. In the formalism we require another function, Duration(s_i, a_j), which returns the duration of an action in a given state. Similarly to cost, duration is often treated as state independent: $\forall_{i,j,k} Duration(s_i, a_j) = Duration(s_k, a_j)$.

Cost-optimizing planning is the problem of finding plans that attempt to minimize cost (or maximize reward) according to a single specified metric. A common class of optimal planners [21, 27] focus on the specific case in which the cost of a plan is simply the sum of the fixed costs of its constituent actions. For these planning problems, the solution sought is a single plan with optimal cost. Optimal planning is an area of automated planning where the generated plans have to be optimal according to an evaluation function. This evaluation typically uses the plan length as the metric to minimize. Some approaches also extend this model and use a context independent, constant cost action model. There is currently no optimal planner which evaluates the plan based on the plan metric composed of metric fluents. Optimal planners have become more powerful in recent years [33] [63] but there remain two limitations on their use in the context of generating pareto-frontiers. Firstly, they are currently designed to work with a simple cost scheme in which each action is assigned a fixed additive cost, which means that they are not useful for solving problems where action costs vary according to the state in which they are applied. Secondly, the restriction to single cost actions makes it much more difficult to use them in multi-objective cases, where the costs of single actions must be computed using an appropriate combination of costs generated in each of the metric dimensions.

If the focus is on the speed, not on optimality, and the user is satisfied with a suboptimal

solution, satisficing planning offers a more practical approach to solving larger instances of problems. Satisficing planning is concerned with quickly finding feasible plans, with less concern about the cost than the efficiency in discovery of the plans. Quality of the plan remains an important property. Although there are no quality guarantees, some planners tend to find solutions close enough to optimal for many applications. For small and simple problems, current satisficing planners often generate optimal or very close to optimal plans.

2.2 PDDL

Planning Domain Description Language (PDDL) [40] is the main modelling language used by the planning community. Across years, many features have been added to PDDL and captured in its different specifications. PDDL is used as the language to describe problems for the International Planning Competition (IPC). A lot of research effort in planning is directed towards the extension of the expressiveness of PDDL. This is reflected in the regular updates to PDDL, such as PDDL 2.1 [16], PDDL+ [15] and PDDL 3 [22], to name but a few. Below, two of the most significant versions of PDDL for this work are presented.

2.2.1 PDDL 1.2

From the outset, PDDL separated the domain description from the problem description. The domain description consists of

- **Domain name** - gives a distinct name to the domain being described.
- **Requirements** - specifies parts of the language actually used in the description such as fluents, numbers, time etc.
- **Object types** - a hierarchy of types of objects used in this domain and any problem file associated with it.
- **Constants** - a list of objects which appear in *all* problems associated with this domain.

- **Predicates** - a list of predicates defining the state propositional variables.
- **Functions** - list of function schemas defining the numeric state variables of the domain.
- **Actions** - a list of actions. Each action has its preconditions, which must be met for the action to be applicable, and effects which become true when the action is applied.

In general action effects can be conditional or probabilistic, however, in this work we only consider non-conditional action effects.

2.2.2 PDDL 2.1

In their specification of PDDL2.1, Fox and Long [16] introduced plan metrics, which are functions of the metric parameters of a planning problem used to evaluate the cost or reward value of a solution plan. Plan metrics are composed of metric fluents. An example expression specifying a metric to minimise is:

```
(:metric minimize (+ (* 4 (energy)) (+ (labor) (* 9 (pollution)))))
```

The format of defining the metric function forces the user of the planning system to explicitly state weights between the metrics. All metrics composing the main metric, which is minimised or maximised, have fixed weights throughout the planning process.

2.2.3 Multiple metrics in PDDL

This work extends the specification of planning problems to allow multiple metrics in the same problem file. This frees the user from defining relative weights between metrics.

An example specification of multiple metrics is given below.

```
(:metric minimize (energy))
(:metric minimize (labor))
(:metric minimize (pollution))
```

Metric functions without weights give greater flexibility to the planner. They also eliminate the issue of a human decision maker making errors of judgement in the weightings affecting the relative quality of the solutions. The objective for a planner faced with multiple metrics is to present the user with alternative solutions trading-off between the different metrics. This can be achieved in various ways and is further discussed in chapter 5.

2.3 Metrics in planning

This section discuses further the use of metric fluents and plan metrics, where these terms refer to Definition 2.1 and Definition 2.2 from page 11.

As mentioned, PDDL 2.1 allows the specification of plan metrics using metric fluents. The plan metric is used to evaluate the quality of the plan and, without loss of generality, we can assume we always use minimisation. An example of a plan metric composed of two metric fluents $(walked)$ and $(fuel\text{-}used)$ is:

```
(:metric minimize (+ (* (walked) 5) (* (* (fuel-used) (fuel-used)) 4))))
```

which is equivalent to the mathematical formula $(5 * walked + 4 * fuel\text{-}used^2)$.

An action can either increment or decrement metric fluents, therefore it could either increase or decrease the plan metric. Although plan metric can be composed of metric fluents using arbitrary aggregation methods, linear combinations of metric fluents are typically used. This limits the practical expressiveness of the plan metric.

2.3.1 Metric solution space

Definition 2.3. A metric solution space is an N dimensional space where each of the dimension is defined by a metric fluent or a plan metric (composed of metric fluents).

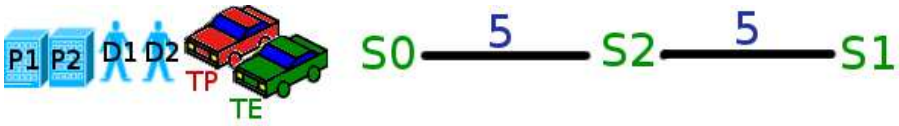


FIGURE 2.1: Example Driverlog problem. The goal is to deliver package P1 to location S1, package P2 to S2 and to make sure driver D1 is at location S1.

Metric solution space, Definition 2.3, is different to the search space and solution space where the former is defined by the set of possible states a planner can explore and the latter is a set of feasible solutions. The metric solution space can be seen as a projection of a solution space onto a metric space. A solution space can be projected onto any metric space. Using this definition we can present a plan in the N dimensional metric solution space as a point, whose coordinates are determined by the values of the metric fluents, or combinations of metric fluents, which define the dimensions.

Consider a problem as depicted in Figure 2.1. This is an instance of a modified Driverlog problem where there are two types of trucks, electric and diesel, two drivers, and two packages at location S0. There are three locations, S0, S1 and S2, and all of them are connected. Table 2.1 illustrates four different plans solving this problem. For the sake of simplicity let us assume that driving a truck between locations consumes the amount of resource (electricity or petrol) equal to the distance between these two locations. Therefore driving between location S0 and S2 costs 5 units of electricity or petrol.

With this assumption Plan 1, from Table 2.1, has the cost of 0 electricity and 10 units of fuel, which we denote as a tuple (0, 10). The other three plans, Plan 2, Plan 3, and Plan 4, have the following resource costs respectively (10, 0), (5, 10), (10, 5). These plans can be graphically represented as points on a metric space as illustrated in Figure 2.2. This is how we construct a graphical metric solution space.

In planning it is common to obtain a concave solution space. Consider a Driverlog domain with an additional electric vehicle, please see Figure 2.1 and Figure 2.2 (Page 18) for an image

TABLE 2.1: Four plans solving a simple Driverlog problem represented in Figure 2.1. Every time an action uses truck TP it uses a diesel truck and each time truck TE is used it refers to an electric truck.

Step	Plan 1	Plan 2
1	(LOAD P2 TP S0)	(LOAD P2 TE S0)
2	(LOAD P1 TP S0)	(LOAD P1 TE S0)
3	(BOARD D1 TP S0)	(BOARD D1 TE S0)
4	(DRIVE TP S0 S2 D1)	(DRIVE TE S0 S2 D1)
5	(UNLOAD P2 TP S2)	(UNLOAD P2 TE S2)
6	(DRIVE TP S2 S1 D1)	(DRIVE TE S2 S1 D1)
7	(UNLOAD P1 TP S1)	(UNLOAD P1 TE S1)
8	(DISEMBARK D1 TP S1)	(DISEMBARK D1 TE S1)

Step	Plan 3	Plan 4
1	(LOAD P2 TE S0)	(LOAD P2 TP S0)
2	(LOAD P1 TP S0)	(LOAD P1 TE S0)
3	(BOARD D1 TP S0)	(BOARD D1 TE S0)
4	(BOARD D2 TE S0)	(BOARD D2 TP S0)
5	(DRIVE TP S0 S2 D1)	(DRIVE TE S0 S2 D1)
6	(DRIVE TE S0 S2 D2)	(DRIVE TP S0 S2 D2)
7	(UNLOAD P2 TE S2)	(UNLOAD P2 TP S2)
8	(DRIVE TP S2 S1 D1)	(DRIVE TE S2 S1 D1)
9	(UNLOAD P1 TP S1)	(UNLOAD P1 TE S1)
10	(DISEMBARK D1 TP S1)	(DISEMBARK D1 TE S1)

representation of the domain and the non-dominated solution space respectively. In this example, when a planner attempts to use two resources simultaneously the solution obtained can not be better than the solution obtained using only one, due to the need of arriving to S1.

Consider a domain with two interchangeable resources, where solutions using N units of the first or N units of the second resource exists. In such domains, although it is not always the case, it is common that a solution using two resources simultaneously incurs a cost of $\frac{N}{2} + cost_1$ of the first resource and $\frac{N}{2} + cost_2$ of the second resource. This is the reason why concave solution spaces are common.

Using a linear combination of metric fluents cannot find all solutions within a concave metric solution space. For example, plans 3 and 4 cannot be found, by an optimal solver, for any linear

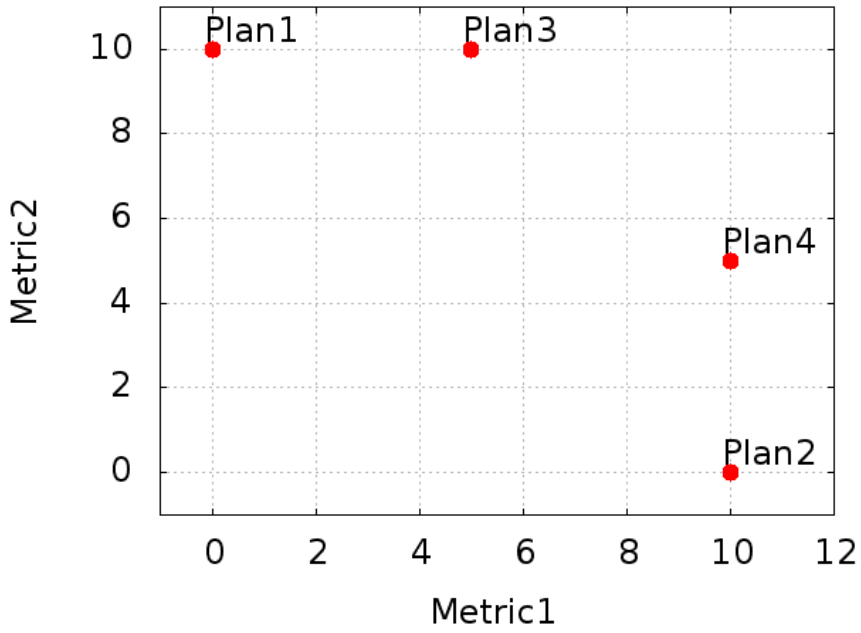


FIGURE 2.2: Metric space with four plans from Table 2.1 solving problem illustrated in Figure 2.1

weighting of *metric*1 and *metric*2.

Further discussion of linear combinations of functions and how the can be used to find solutions in convex and concave metric solution spaces is shown in Chapter 5. In planning, we are not able to compute a place in the metric space where a solution should be without going through the planning process and obtaining a solution first. Therefore, in planning a linear combination of metric fluents is usually sufficient, which is also shown in the result section for Chapter 5.

2.3.2 Metrics and plan length

Plan quality is often closely linked to plan length, but the interactions can be subtle. The classes of interaction can be used to classify current planning problems based on their properties. Although, in terms of syntactic representation, all of the classes can be expressed using the same PDDL constructs, each of the classes represent a significantly different challenge to planners.

Radzi [50] distinguishes different interactions between plan lengths and plan metrics. The four classes, equivalent to those presented in Radzi's thesis, are presented below. In the following we use π and $\pi;\sigma$ to stand for executable sequences of actions, where the latter is a concatenation of two sub-sequences, $c(\pi)$ as the cost of π and $|\pi|$ as the length of π.

1. **Strictly straightforward metric function** $\forall\pi_1\pi_2 \cdot |\pi_1| \leqslant |\pi_2| \Leftrightarrow c(\pi_1) \leqslant c(\pi_2)$
2. **Straightforward metric function** $\forall\pi_1\pi_2 \cdot |\pi_1| < |\pi_2| \Rightarrow c(\pi_1) < c(\pi_2)$
3. **Semi-straightforward metric function** $\forall\pi\sigma \cdot c(\pi) \leq c(\pi;\sigma)$
4. **Expressive metric function** $\exists\pi\sigma \cdot c(\pi) > c(\pi;\sigma)$

The first class of interaction occurs when by minimising the plan length we also minimise the plan metric. By obtaining the shortest possible plan we also obtain the best quality, optimal, plan.

The second class of interaction is similar to the first. The best quality plan is also the shortest. However, within this class, finding the shortest plan is not a guarantee of finding an optimal plan. There may exist two plans of equal length but with different quality. Therefore it is not sufficient to find the shortest plan to guarantee optimality. The optimal plan cannot be longer than the shortest plan.

The shortest solutions for domains in the third class do not guarantee to be optimal. Although the metric is monotonic, which means that the shorter plans tendt to be better quality, there is a possibility of obtaining longer plans, using low cost actions, which are better quality than some very short plans which utilise expensive actions.

The fourth, *expressive metric function* class of domains, creates more interesting challenges as it is possible for the plan to lower the cost by applying additional actions.

Based on Radzi's work the current planners can be grouped by the class of metrics which they can minimise in an informed way. For the classification, we abbreviate the list by merging Radzi's first and second class. From now on we use the following metric and plan length interaction.

Definition 2.4. We say a plan metric, c, is:

1. *Strictly length-correlated* if $\forall \pi_1 \pi_2 \cdot |\pi_1| < |\pi_2| \Rightarrow c(\pi_1) < c(\pi_2)$.
2. *Monotonic* if $\forall \pi \sigma \cdot c(\pi) \leq c(\pi; \sigma)$.
3. *Non-monotonic* if $\exists \pi \sigma \cdot c(\pi) > c(\pi; \sigma)$.

While some planners optimise quality only when the plan metric is strictly length-correlated, only metric sensitive planners can effectively deal with monotonic metrics. There are currently no planners that deal effectively with non-monotonic metrics.

In addition to the above classification, planners can be divided depending on their ability to handle context-dependent action costs. In this research we focus on planners which can effectively reason with domains with such cost models.

2.4 Plan distance

It is important to be able to compare plans based on their relative qualities, but also to assess differences between plans. This section describes various methods for measuring distance between plans. Some of these methods are successfully used in generation of sets of distinct plans [45, 46], where the distances are measured in terms of actions, causal links, or states visited. At the end of this section we propose to measure the plan difference in terms of the metrics by which the plans are evaluated in a given domain. The details of this method are presented below.

2.4.1 Action distance

Action distance between plans is defined based on the similarity measure of the sets of actions contained within those plans. Actions in the plan can be considered as ordered sequences or unordered sets. When dealing with sequences of actions, the score depends on how close to each other the sequences are. This can be measured by calculating the hamming distance (Definition 2.5) or Damerau–Levenshtein distance (Definition 2.6) between them.

Definition 2.5. Hamming distance between plans π_1 and π_2 of equal length is the number of positions at which the corresponding actions are different. In another way, it measures the minimum number of substitutions required to change one plan into the other.

Definition 2.6. Damerau–Levenshtein distance between plans π_1 and π_2 is based on the Damerau distance [10] and Levenshteins edit distance [37] and can be defined as the amount of operations required to transform π_1 into π_2, where the allowed operations are: insertion, deletion, or substitution of a single action, or a transposition of two adjacent actions.

Alternatively, for all actions a distance can be calculated between the positions of occurrences of the action in respective sequences, or a certain values if an action appears only in one. When plans are considered as sets of actions the distance might be calculated as the Jaccard index, $\frac{\pi_1 \cap \pi_2}{\pi_1 \cup \pi_2}$, for plans π_1 and π_2

The appropriate method is always stated whenever this measure is used and is denoted as d^a. One of the benefits of this method is that no additional knowledge of the domain structure is required.

2.4.2 States visited distance

In this measure the ordered sequences of states visited by the execution of plans are compared. This distance is denoted as d^s. Similarly to the action distance, comparison of the sequences

of states can be defined in a number of ways, and the same methods are applicable here. One improvement, compared to the action distances, is to consider the similarity of states at corresponding positions in two sequences. Since states contain more information than actions, the similarity measure between them is more informed than the similarity score between two actions. When two plans arrive at a goal via similar states, they can be considered similar. The similarity of states can be measured using the hamming distance (Definition 2.5) or Damerau–Levenshtein distance (Definition 2.6) or one of the set similarity measures such as the Jaccard index or the bag of words model. These measures, when aimed at similarity of states, are applied to facts true in the states being compared.

2.4.3 Causal links distance

A causal link is a tuple (a_i, p, a_j), where $a_i, a_j \subset A$ and $p \subset P \cup V$ and where $p \subset Add(a_i)$ and $p \subset Precondition(a_j)$. This measure, denoted by d^{cl}, represents the difference in a causal structure of one plan comparing to the other. A causal link gives information about which facts enable an action and which facts are achieved by an action. Plans are expanded into sequences of causal links representing how each action contributes to the goal. The distance measure using causal link sequences can be computed in a similar way to the two previous measures.

2.4.4 Metric distance

A more practical distance between two plans is defined based on their relative position in a metric space. This distance is equivalent to an euclidean distance between the plans on the selected metric space. Definition 2.7 presents a more formal definition of this metric.

Definition 2.7. Metric distance between plans π_1 and π_2 is a Euclidean distance in the space described by the metrics Θ_i, i=1, ..., n.

$$d^m(\pi_1, \pi_2) = |\pi_1 \pi_2| = \sqrt{\Sigma_{i=1}^{n} (\Theta_i(\pi_2) - \Theta_i(\pi_1))^2}$$

This distance depends on the set of plan metrics selected. In order to calculate this distance, the coordinates on the metric space must be known. These are calculated using the plan metrics defined by the user. Therefore the choice of the plan metrics determines the metric distance between plans. It is often the case that for a choice of plan metrics, plans which appear the same in terms of metrics are different in terms of actions, states or causal links.

Due to the fact that the user specifies the plan metrics which are of interest and these measures are used to evaluate the difference between plans, plans considered the same represent trade-offs between resources that are not interesting to the user. At the same time, if a plan affects plan metrics which the user is interested in, the plans appear distinct.

2.5 Multi-objective domains

Previous sections present the components for a planner. These components include the modelling language, the concept of metrics and plan quality measures. In this section a selection of multi-objective domains is presented. These domains act as benchmark tests to illustrate the difficulty and challenge in multi-objective planning with metrics.

By multi-objective domains we mean domains where the solution plan is evaluated using a set of plan metrics. Although many current benchmark domains contain metrics, evaluating the solution using these metrics does not yield interesting results. The main reason for that is the coupling between the plan length and plan metric function value. Also, the current benchmark does not offer interesting trade-offs in arriving to the goal by means of different metrics. This usually means that there is a single plan minimising all metrics at the same time, without any interesting trade-off. Domains presented here all offer multiple paths to the goal, where the paths significantly differ in terms of the plan metric function value. The domains Production and Bread were introduced by Radzi [50] and Driverlog is a modified IPC Driverlog domain.

2.5.1 Driverlog

The Driverlog domain [38], used in the planning competition, models a simple transportation problem. The domain contains the following objects: truck, driver, package and location. Drivers can walk between locations and a problem description typically contains one additional intermediate location which must be used when walking. A truck can drive between locations only when a driver is driving the truck and the locations are connected. A package can be loaded and unloaded from a truck. A usual goal is to deliver certain packages to a certain location and to assure that drivers are at specific locations.

The domain is also available in a metric version. The metric version of the domain walking and driving between cities is associated with a cost. The planner is required to optimize the linear combination of these two costs. Both of them are correlated with the plan length, and the package must be transported using a truck. This means there is no alternative routes of achieving the goal but to load the packages onto one of the trucks and drive it to the goal.

The Driverlog domain used in experiments is the standard benchmark but differentiates electric and diesel trucks in order to create a trade-off between diesel and electricity consumption used for transportation of packages. The planner can choose whether to transport packages using electric or diesel vehicles, and therefore the resulting plan can exploit the trade-off between these resources. For example, in cases where electricity is an expensive resource, expressed through application of a high weighting on the metric fluent representing the electricity used, the planner should favour plans using diesel trucks.

The Driverlog domain with context dependent action cost, used in this research, is available in Appendix F.

2.5.2 Driverlog metric

Driverlog_metric is a further modified version of the Driverlog domain. For the purpose of creating a domain where the cost and plan length are not strictly length-correlated, as in Definition 2.4, the following modification is applied: three of the locations in the domain, $s0$, $s1$ and $s2$, are connected by very short routes with two intermediate steps. These short steps allow the planner to take shorter routes, which consumes less resource, and at the same time requires more actions. Instead of a single drive action, a truck must drive through two intermediate locations using three drive actions. This change tests whether a planner is metric sensitive and uses cheap actions, as opposed to relying on correlation between plan length and cost.

2.5.3 Production

The aim in the Production [50] domain is to obtain a certain amount of materials and ready products in stock. There are various ways of creating the same product. In order to build a product, we need to produce necessary materials for it. Creating materials can also be achieved in many ways, for example we can obtain materials from “recycled material” in a recycling process or from “raw material” by processing it. The planner has flexibility to use various methods to arrive at the same goal which differs in labour, hazard or machine-cost values. These three functions are our primary metric functions to minimise, using different weightings.

2.5.4 Bread

The Bread [50] domain describes a process of baking bread and buns. We start with flour which we turn into a mix. The mix is used to create a dough. Dough is either created by hand or using a machine. Using the machine increases energy consumption and the machine needs cleaning, but can create twice more dough comparing to doing it by hand. The next stage is to make a bun or a bread from the dough. From the same amount of dough we can form either two loaves of bread or five buns. They can be then baked using either an electric oven or on

charcoal. An electric oven uses one unit of energy and charcoal increases the pollution by one unit. An electric oven can bake ten buns or four loaves while charcoal only two buns or two loaves. Our metric functions to minimise are energy, labour and pollution.

2.6 Relaxed Planning Graph

Many current planners use a Relaxed Planning Graph (RPG) as the basis for heuristic calculation. This approach has proven to work remarkably well for the purpose of generating informed heuristics very fast. We now present an example RPG for a sample problem.

RPG contains interleaved Fact and Action layers, where fact layers contain propositions which are true at a given state. In metric planners, the fact layer also contains estimated values, sets of values or intervals of metric fluents. Action layers contain actions which are applicable in a "state" described by the previous fact layer. The first layer of the Relaxed Planning Graph is constructed from the state being evaluated. The first action layer contains all actions applicable in the first fact layer.

After this collection of initial steps, every next fact layer is a union of the previous fact layer with all positive effects of all preceding action layer effects. This method ignores delete effects.

For planners handling metric fluents, the most common approach is to keep lower and upper bounds on the value of each metric. Every time a metric increases, its upper bound is also increased by the same value. Similarly, every time a metric is decreased its lower bound is also decreased by the same value. This ensures that the value of the metric, no matter which actions are selected from the relaxed planning graph, is always between the computed lower and upper bounds.

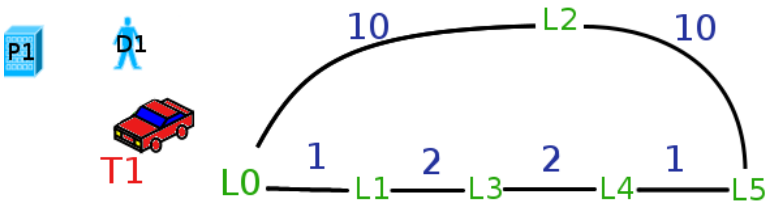


FIGURE 2.3: Diagram represents an example problem for a Driverlog domain.

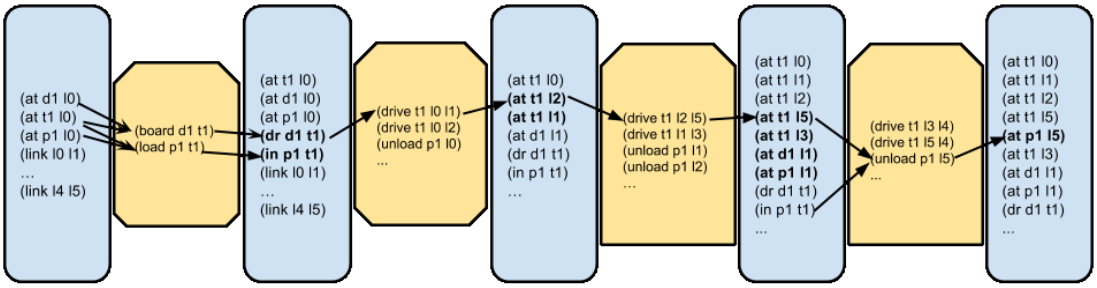


FIGURE 2.4: Diagram represents extract of RPG built for problem represented by Figure 2.3

2.6.1 Example construction of the RPG

Figure 2.3 represents a start state of a problem for Driverlog domain. In this state, at location L0 there is a truck, t1, a driver, d1, and a package, p1, which have to be delivered to location L5. There are two possible roads, via L2 and via L1. The road via L2 is much longer in terms of the distance, and the road via L1 is shorter in terms of the distance, but requires more actions. The RPG-based heuristic constructs an RPG as shown in Figure 2.4.

We can see that the path via L1,L3,L4 would not be considered as the path via L2 achieves the goal earlier in the RPG, therefore action (drive d1 t1 L0 L2) is the most preferred action in the initial state, which will then result in the entire plan going via L2.

2.6.2 Temporal Relaxed Planning Graph

In this section we briefly discuss the Temporal Relaxed Plan Graph (TRPG) [9] The idea in the temporal setting, which is similar to the construction used in Sapa [14], is to extend the plan graph, during construction, with applicable action *start* points, queueing their end points

to occur at the corresponding action duration interval after the start. Once all applicable action starts have been identified and applied, arriving at a fixed point for this stage, the *earliest* queued action end point is applied and then the process is repeated, until this queue is empty (or the goals are achieved). Critically, each layer of the TRPG is labelled with the time at which it is reached. This means that the makespan of the relaxed plan can be identified directly within the TRPG and this leads to an informative temporal heuristic.

2.7 Satisficing planners

A selection of state-of-the-art satisficing planners, which are capable of reasoning with metric fluents, is described in this section. Reasoning with metrics is a crucial requirement in terms of this research. Satisficing planners, due to their speed, offer a very promising approach to solving larger problem instances. Many satisficing planners can generate high quality plans for domains containing metrics. Some planners aim at generating high quality solutions by minimising plan length and do not support metric fluents. Those planners generate the same plan for two instances of a problem, which differ only in the plan metric. Therefore they are not interesting for comparison with planner which aim at reasoning with metrics. In some cases [26, 53] planners allow a special predefined metrics fluent to be used as a cost. Incrementing this metric fluent is used to increment the cost, where the increment on the metric is equal to the increment of the cost of the plan. This approach is analogous to using constant cost as it also ignores all other metrics.

Our work focuses on providing the flexibility for the user of the planning system to define functions against which the plan is evaluated. Therefore the fixed action cost approaches are not applicable, since the planner must generate good quality plans depending on the plan metric. The candidates among satisficing planners which are interesting from the perspective of this research are presented below.

2.7.1 LPG-td

LPG-td [23] is a metric modification of the LPG [24] planner. LPG is a local search, stochastic planner. It creates its search space based on a graph with interleaved proposition and action layers called a numerical action graph (NAG). Its heuristic consists of two elements; search cost and execution cost, where search cost is an estimate cost to resolve all inconsistencies created by inserting a new action. It is estimated by solving a relaxed NAG. Execution cost is the total cost of executing actions in the plan and it represents plan quality. There are two weights on these two components which allow trade-off between finding a solution quickly or searching for a good quality solution, depending on the need and constraints. LPG has been adapted to generate sets of plans [45,46]. The adaptation is to use the Integrated Convex Preference (ICP) measure, shown in Definition 2.8, inside its heuristic, instead of the standard execution cost. The ICP is a convex combination of plan cost and plan makespan.

Definition 2.8. Integrated Convex Preference (ICP) is a special case of Integrated Preference Function (IPF) [6]. For a set of plans $\pi \in \phi$ and cost and makespan of plans denoted as c_π and t_π respectively. The ICP of the set of plans ϕ, consisting of k plans, and parameter w is calculated as:

$ICP(\phi) = \sum_{i=1}^{k} \int_{w_{i-1}}^{w_i} h(w)(w \times t_{\pi_i} + (1-w) \times c_{\pi_i}) dw$

Where h(w) is a probability distribution for the parameter vector w such that $\int_w h(w)dw = 1$.

In this work we often use different configurations of LPG. The main option with which we experiment is -n <number>. This option allows LPG to generate a solution and if <number> is greater than 1 it then attempts to improve the solution <number>-1 times.

2.7.2 MetricFF

MetricFF [28] is an extension of the FF planner [30]. As an extension to a delete relaxation, removing all negative effects of actions, it handles linear numerical effects by relaxing the

metric variables to their lower and upper bounds. Therefore, if at some point in the RPG construction, x>2 becomes true, it remains true for all of the following layers. When the goal is reached in the RPG, the planner extracts the relaxed plan backwards from the last RPG layer. Although its heuristic considers numeric effects, in practice it aims at minimising the plan length. This is because storing upper and lower bounds on the metric fluents is not sufficient to reason about their actual values, so manipulation of resources is not optimised.

A more recent version of MetricFF [29] is enhanced with an A*-epsilon [49] search strategy to better handle the additive metric cost.

2.7.3 CBP

Cost-Based Planner (CBP) [19] is a deterministic, any-time planner which uses multiple heuristics. It implements a cost-oriented heuristic by altering an RPG expansion. An additional list of opened applicable actions, *OpenApp*, is stored. When creating the RPG, *OpenApp* contains all actions which preconditions are satisfied in the latest fact layer. The next action layer is created from the cheapest actions from the *OpenApp* and not, as it is done in the standard version of RPG, from all of the applicable actions in the latest fact layer. This results in an RPG construction algorithm based on Dijkstra algorithm, as opposed to the breadth-first search for the standard RPG.

The planner uses look-ahead state [60] to improve its performance. The look-ahead states are obtained by applying a selection of actions from the relaxed plan, extracted from the RPG. The gain from applying this technique depends on the similarity of the relaxed plan extracted from the RPG and the actual plan satisfying the goal condition.

The search algorithm used is a weighted Best-First Search (BFS) with Branch and Bound (B&B). The heuristic evaluation for the weighted BFS algorithm is $f(x) = g(x) + \omega h(x)$. The B&B algorithm finds a solution, and then tries to improve it. It prunes states based on the $g(x)$ value only because the heuristic evaluation is inadmissible. The nodes are expanded based on

three lists. First an *open* list is expanded, and every time a node from that list is expanded its look-ahead state is added back to *open*. Helpful actions from the relaxed plan, stored with the state, are added to $sec-ha$. Non-helpful successors of the state are added to the $sec-non-ha$. When *open* list is empty, states from $sec-ha$ are used. When $sec-ha$ is empty, states from the $sec-non-ha$ are used.

This strategy proved to work well on domains with constant action costs.

2.7.4 POPF

POPF2 [9] is a forward search planner which exploits some partial ordering of actions to avoid searching all orderings. That means it does not enforce a strict total ordering on actions before the final stage of planning. For all facts and variables it keeps a list of propositions it has to support in order to execute the plan. While expanding a node, which is a partial-order plan, it adds actions and creates new partial-order plans accessible from the current one. POPF2 attempts to find minimum makespan plans. It uses a Simple Temporal Network (STN) to handle temporal constraints between actions and schedule them in a feasible way. It handles metric fluents, as for PDDL 2.1, using methods based on MetricFF.

2.7.5 LPRPG

LPRPG [8] uses a relaxed planning graph (RPG) heuristic combined with linear programming (LP) methods. It solves a number of LPs for every decision it makes to calculate bounds on resources and to improve its numeric reasoning. Thanks to solving the LP, LPRPG has more precise information about bounds on resources than other planners and therefore is designed for use in domains with numeric resource flows. Its heuristic calculates the upper and lower bounds on metric fluents much more precisely than other planners, through the LPs it constructs to tie together resource production and consumption, allowing it to more efficiently solve problems involving resources than many other planners.

2.7.6 YAHSP

YAHSP (Yet Another Heuristic Search Planner) [61] is a satisficing STRIPS planner. It achieved 2nd place in the suboptimal track of the IPC 2004 [38]. Its heuristic is built in a similar way to that of MetricFF. YAHSP also uses a RPG and it extracts information about helpful actions. Actions that appear in the first action layer of the RPG, and also are part of a relaxed plan, are considered helpful. These actions are used with higher priority than others in search. In addition to helpful actions, YAHSP uses look-ahead states [60] to speed up plan discovery. A best-first search strategy is used to favour states achieved by applying helpful actions.

This planner has been adapted by a multi-objective planner, MO-DAE, as a sub-solver. MO-DAE is further discussed in Section 5.1.

2.8 Pareto frontier

Typically planners output a single plan for a given problem. In multi-objective domains, as described in Section 2.5, a single solution is no longer sufficient. The new task for the planner is to generate a set of solutions which illustrate the different trade-offs possible within the domain.

For simplicity, and without the loss of generality, let us only consider minimising plan metrics. When minimising a single objective function, it is easy to determine a better solution by simply calculating the plan metric function at the end of the plan execution. Having the plan metric value, the better plan is the one with a smaller plan metric value, therefore for two plans π_1 and π_2, π_1 is better if $\theta(\pi_1) < \theta(\pi_2)$ for a plan metric function θ.

In multi-objective domains the relations between plans are more complex. We say that a plan π_1 dominates π_2 if it is better in all plan metrics as in Definition 2.9.

Definition 2.9. Plan π_1 **dominates** π_2 for a set of plan metrics $(\theta_1, \theta_2, ...\theta_N,)$ if:
$\forall_{i=1..N} \theta_i(\pi_1) < \theta_i(\pi_2)$.

We also say that a plan π_1 weakly dominates π_2 if it is better in at least one plan metric and no worse in all others as in Definition 2.10.

Definition 2.10. Plan π_1 **weakly dominates** π_2 for a set of plan metrics $(\theta_1, \theta_2, ...\theta_N,)$ if:
$\forall_{i=1..N} \theta_i(\pi_1) \leqslant \theta_i(\pi_2)$ and $\exists_i \theta_i(\pi_1) < \theta_i(\pi_2)$.

A set of all non-dominated plans is called a pareto frontier (PF). A pareto frontier typically denotes a set of non-dominated and optimal plans.

When using satisficing planners optimality is not guaranteed. The goal of the planning process is to generate a good approximation of the PF. What we mean by good approximation is discussed in Chapter 5.3. In order to approximate the PF we introduce a notion of Approximation of Pareto Frontier in Definition 2.11:

Definition 2.11. Approximation of Pareto Frontier (APF) is a set where any element does not weakly dominate any other element.

This definition does not speak about the quality of the set, and how far does it lie from the actual optimal PF. This is further discussed in the Section 5.3 about the quality of APFs.

Chapter 3

Metric sensitivity

In this chapter we present a definition of metric sensitivity [55] and discuss its meaning, as well as a method for quantitative evaluation of the metric sensitivity of planners. Current state-of-the-art planners are discussed and an evaluation of their metric sensitivity is presented. The results show that metric sensitivity is a rare, although desirable, property of satisficing planners. We go on to consider the impact of stochasticity on metric sensitivity and whether we can enhance metric sensitivity by adding stochastic behaviour to deterministic planners.

Metric sensitivity is a crucial element in the weighted sum approach to planning with multiple objectives. This method, described in Chapter 5, needs to have some control over where, in the metric space, the planner generates solutions.

3.1 Example

We now present a simple concrete example problem that illustrates some of the issues involved in finding high quality plans under different metric functions. Suppose we want to transport two packages from location $L0$ to location $L5$. We have three vehicles available, one electric, $Te1$, and two diesel, $Tf1$ and $Tf2$, and two drivers available. The amount of diesel or electricity used

by a vehicle is equal to the distance driven multiplied by the *square* of the loaded truck weight (number of packages plus one) and its resource (diesel or electricity) consumption (we use 1 diesel or 1 electricity for this example). This problem is represented in Figure 3.1. Assume we are using the metrics within the weighted metric function, namely, (electricity-used) and (fuel-used).

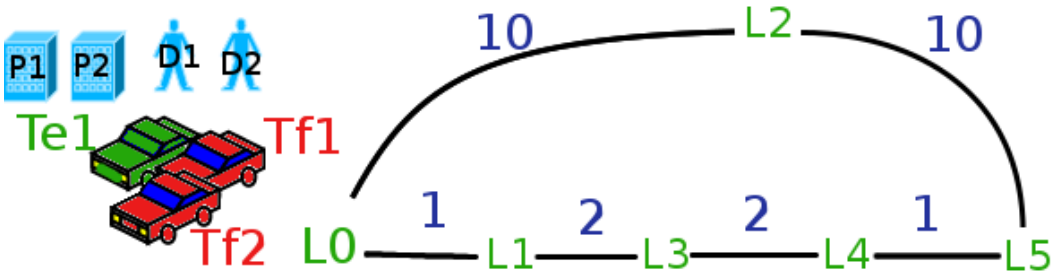


FIGURE 3.1: A simple problem with different vehicle types to transport packages $P1$ and $P2$ to $L5$.

Five plans solving this problem are summarised below:

1. Load both packages into $Tf1$ and then drive them to $L5$ via $L2$ using driver $D1$. Cost: 180 diesel, 0 electricity.
2. Load both packages into $Tf2$ and drive them to $L5$ via $L2$ using $D1$. Cost: 180 diesel, 0 electricity.
3. Load both packages into $Tf1$ and drive to $L5$ via $L1$, $L3$ and $L4$, using $D1$. Cost: 54 diesel, 0 electricity.
4. Load both packages into $Te1$ and drive to $L5$ via $L1$, $L3$ and $L4$, using $D1$. Cost: 0 diesel, 54 electricity.
5. Load one package into $Tf1$ and one into $Tf2$ and drive both to $L5$ via $L1$. Cost: 48 diesel, 0 electricity.

The first two plans illustrate the difficulty in comparing plans based on the number of different actions: these plans are considered quite different, because they use different vehicles, but are identical with respect to the metrics (so separated by distance 0 according to Definition 2.7).

They do not offer interesting alternatives in an APF. A similar problem arises for plans using the alternative driver.

Plans 1, 3, 4 and 5 are all different, both qualitatively and quantitatively (by Definition 2.7). It is clear that the optimal cost plan, under any combination of metrics, will involve using the shorter path. The fact that this uses more actions is a problem for many planners, which attempt to minimise plan length as a proxy for the plan quality. Plan 5 dominates plans 1, 2 and 3, while plan 4 does not dominate any and is not dominated by any other plan presented above.

Finally, using a single metric combining fuel cost and electricity cost with respective weights of 9 and 8, the optimal plan uses one diesel truck and the electric truck, each carrying one package (cost 24 diesel and 24 electricity, with total weighted cost of 408, while both plans 4 and 5 have weighted cost of 432). This example illustrates how interesting choices arise according to the trade-offs between resources being used depending on the plan metric.

3.2 What is metric sensitivity

Metric sensitivity is the ability of a planner to generate good quality solutions for the metric defined in the problem description. This requires the planner to respond to changes in the metric by generating good quality plans under alternative metrics.

We call a planner metric sensitive if, presented with two different (cost) metrics, m_1 and m_2, it produces different plans: π_1 and π_2 for the same problem instance, such that $m_1(\pi_1) < m_1(\pi_2)$ and $m_2(\pi_1) > m_2(\pi_2)$. In practice, a metric sensitive planner will not always be able to find two different plans that are each better under their respective metrics. Therefore we need to relax the above strict inequalities to: $m_1(\pi_1) \leqslant m_1(\pi_2)$ and $m_2(\pi_1) \geqslant m_2(\pi_2)$.

Definition 3.1. A planner is metric sensitive if for all domains, for all problems, for every two metrics m_1 and m_2, two plans, π_1 and π_2 generated using metrics m_1 and m_2 respectively, the following holds: $m_1(\pi_1) \leqslant m_1(\pi_2)$ and $m_2(\pi_1) \geqslant m_2(\pi_2)$.

The impact of this property on a planner for the purpose of generating a frontier of plans is to increase the control over where, for a given plan metric, a plan is found in the metric space. Suppose we have a problem where we can trade-off labour and financial cost metrics and that a solution, which is cheap in terms of financial cost but requires high labour, is known. In order to generate a set of plans capturing the trade-off between the two resources a solution with lower labour cost is required. A metric sensitive planner, given a plan metric with high weights on labour, should generate a solution that is cheaper in terms of labour, provided such a solution is feasible. Following the property above, it would generate no worse solution in terms of the amount of labour required.

For stochastic planners, metric sensitivity as defined in Definition 3.1 sometimes does not apply, even though the planner does respond to the change of the metrics. This behaviour can be seen in Figure 3.3 where sets of plans generated for the same metric function are present. The weighted metric function is $(\alpha * (fuel - used) + (1-\alpha) * (electricity - used))$, but in the actual experiment, to avoid floating point numbers, we use the integer part of $10 * \alpha$. As can be seen for this stochastic planner, for a given two weights α_1 and α_2 it is possible to find plans from the set Π_1, generated using α_1, which are better, in terms of metric defined by weights α_2, than some of the plans from set Π_2, generated using α_2. This means that we can find cases that contradict the metric sensitivity of the planner. However, we can clearly see from the results that the planner does respond to the change of the metric. Therefore the definition of metric sensitivity for stochastic planners must be extended. We propose the use of centroids of clouds of plans as in Definition 3.2 and to extend the definition of metric sensitivity, Definition 3.1, to stochastic metric sensitivity as in Definition 3.3 which can be applied to stochastic planners.

Definition 3.2. Metric Centroid of a sets of plans, $\pi_j \subseteq \Pi$ where N is the number of metrics, $m_i(\pi)$ is the value of plan π under the i-th metric, and $\overrightarrow{m_i}$ is a unit vector towards the i-th metric, is N-dimensional vector $\overrightarrow{\pi}$ computed as: $\overrightarrow{\pi} = \frac{\sum_{j=1}^{N} \sum_{i=1}^{|\Pi|} m_i(\pi_j) * \overrightarrow{m_i}}{|\Pi|}$.

Definition 3.3. A stochastic planner is metric sensitive if for all domains, for all problems and any two metrics m_1 and m_2 under which it generates the following sets of plans, Π_1 and Π_2 respectively. Two sets of plans have centroids π_1 and π_2 respectively. The metric cost of the centroids of the plan sets satisfies: $m_1(\pi_1) \leqslant m_1(\pi_2)$ and $m_2(\pi_1) \geqslant m_2(\pi_2)$.

The centroid of the set of plans, Definition 3.2, in the metric space can be calculated as an average value of each of the metrics for each of the plans from the set.

The definition for metric sensitivity presented above is very strict. In practice it is useful to also determine whether a planner exhibits metric sensitive behaviour. For this purpose we relax this definition and propose the following:

Definition 3.4. A planner exhibits metric sensitive behaviour if, when faced with problem instances containing different plan metrics, it sometimes generates better plans in response to the change of the metric. Formally: $\exists_{\pi_1,\pi_2 \in \Pi} \exists_{m_1,m_2-metric} m_1(\pi_1) < m_1(\pi_2) \wedge m_2(\pi_1) > m_2(\pi_2)$ where π_1 is generated for the planning metric m_1 and π_2 is generated for the planning metric m_2.

The Definition 3.4 can be used to discuss properties of planners which do not exhibit full metric sensitivity as in Definition 3.1 and 3.3 but are promising candidates for generating plans in response to the plan metric.

3.3 How to measure metric sensitivity

We have shown how to determine whether a planner is metric sensitive. Now we focus on a method to qualitatively assess the *degree* of metric sensitivity in comparison with other planners.

The method we propose to use to evaluate metric sensitivity is based on the scoring metric used in recent International Planning Competitions. After a set of planners is run on a problem instance derived from a multi-objective problem by combining the metrics into a single weighted sum, the solutions are gathered and compared with each other. We denote the cost of plan π_i under metric m as Θ_i. We only compare plans using the same metric and for simplicity we omit metric indexes. For each weighting scheme we calculate the value of the best plan across all of the planners, Θ_{best}, and then we assign a score to each planner, i, producing a plan for this problem with value Θ_i, using the relative metric sensitivity with relation to the best plan found for the particular problem and plan metric.

Definition 3.5. Relative Metric Sensitivity Score is defined as $\Theta_i = \dfrac{\Theta_i - \Theta_{best}}{\Theta_{best}}$.

For the planner which generated the best solution, the value of Θ_i is 0, and it grows with the distance from the best solution found. To avoid division by 0, for the cases where $\Theta_{best} = 0$ we take Θ_i as the score, which is equivalent to subtracting 0 and dividing by 1.

For comparison when a planner does not generate a solution for a specific weighting for a problem the results are discarded for this weighting.

Generating multiple versions of the same problem, by using different weighted metric functions, allows confirmation that a planner does not perform best on a specific instance of the problem and metric as a side effect of its other properties. For example, sometimes the best plan under a given metric might be found by minimising plan makespan. In other cases it is possible that best solutions to a problem are found for a given metric function by chance. However, when presented with a different metric, the planner might generate only poor plans because, it tends to search in the areas where plans happen to be of high quality in terms of the former metric, but are poorer in terms of the second metric. By not using different weightings we could be biased towards a planner which is metric insensitive but, due to its design or other properties, performs well on a subset of weightings.

In this work we typically use solutions to the following simplex as weights: $\sum_{i=o}^{N} \alpha_i = 10$ where all weights α_i are required to be integers. This yields 11 weights for two dimensional problems, namely (0,10), (1,9), (2,8), (3,7), (4,6), (5,5), (6,4), (7,3), (8,2), (9,1), (10,0). For three dimensional problems it yields 66 weights: (0, 0, 10), (0, 1, 9), ..., (10, 0, 0).

3.4 Current planners

A selection of current state-of-the-art planners presented in section 2.7 is further discussed and evaluated in this section. The selection was made based on the potential ability of each planner to exhibit metric sensitive behaviour. The section first focuses on optimal planners as solving a problem optimally must also mean responding to the change of the metric function. Before a final conclusion, we discuss properties of planners which might affect their metric sensitivity and present results showing which planners are and which are not metric sensitive.

3.4.1 Optimal planners

The problem of generating high quality plans is central to the population of APFs as defined in Definition 2.11. Optimal planners, by definition, find plans with minimum costs. Using different weights should give us optimal plans in different areas of the search space while optimising the cost. These plans merged together should give us the pareto frontier of plans. Therefore, optimal planners seem to be good candidates for that task. Below we present some state-of-the-art optimal planners properties and some of their current application relevant to metric sensitivity.

Optimal planners have become more powerful in recent years [33] and have been successfully applied to speed up the process of finding optimal plans when preferences change, using optimal plans already generated for alternative sets of preferences [63]. This has been applied to a personalised vehicle routing problem, where drivers can express preferences about the route. This approach scales well for this particular path planning problem.

Although the above examples look promising, there remain some limitations on the use of optimal planners in the context of generating pareto frontiers. Firstly, they are currently designed to work with a simple cost scheme in which each action is assigned a fixed additive cost. This means that they are not useful for solving problems with state dependent action costs, which are the main focus of this thesis. Secondly, the current domain independent state-of-the-art optimal planners are impractical for solving large instances of domains, or domains with complex cost models.

Currently many planners do not allow metric fluents. In order to simulate action costs, these planners allow a single metric state variable, total-cost. The increment of total-cost is considered to be the cost of an action. This approach is equivalent to using constant cost, but instead of explicitly assigning the cost to an action, it is implicitly assigned by the increment effect of actions on the total-cost. Since, in such domains, there exists only one metric fluent, total-cost, the plan metric is fixed. While optimal planners have improved in performance, satisficing planners offer a more practical approach to the generation of the large sets of plans required to populate an APF.

3.4.2 Satisficing planners

Among satisficing planners there are some that are very promising in terms of being metric sensitive. Satisficing planners, due to their speed, are able to solve larger problem instances and quickly explore different areas of the search space. Many satisficing planners can generate high quality plans for domains containing metrics. Many of them do it as a side effect of optimising plan length, and therefore only work for domains with plan metric strictly length-correlated. In this section we present an experiment which aims at determining which of the satisficing planners are truly metric sensitive. Planners described in Section 2.7 are tested for their metric sensitivity. After a brief description of the experiments carried out, further sections present the

results and discussion of metric sensitivity and properties influencing metric sensitivity of the planners.

3.4.2.1 Metric sensitivity experiment

A metric sensitive planner should respond to the change of a metric, therefore in this experiment all planners are run on the same planning problem with a different metric function. This will indicate whether a planner takes the metric into consideration when planning. A problem is selected from Driverlog numeric domain described in Section 2.5.2 with both diesel and electric trucks, and a series of 11 problem instances is generated. This domain contains context dependent action costs. Such a choice of a domain tests not only for metric sensitivity, but for metric sensitivity in a richer cost models. Each problem instance differs from the others only in the metric function the planner has to minimise. This metric function is composed of two metrics, fuel-used and electricity-used. The metric is a weighted sum of these two, as described earlier.

We run each of the planners, MetricFF with optimisation mode on, LPRPG, POPF2, CBP and LPG in configuration with -n 3. All of them are described in Section 2.7.

The results are present in Figure 3.2. Results suggest that MetricFF, LPRPG, CBP and POPF2 are not metric sensitive as each of them generates a set of 11 solutions where each has the same plan metric values. This is expressed by all 11 plans lying in the same place on the metric space. It is important to note that the plans might differ under other distance metrics like actions used from Definition 2.5. This means that they do not change their behaviour with the change in the plan metric. The reason for this behaviour is the fact that these planners ignore the context-dependent plan metrics when searching for the solution. We suspect that experiencing metric insensitivity from MetricFF and CBP is mainly caused by the presence of context-dependent action costs.

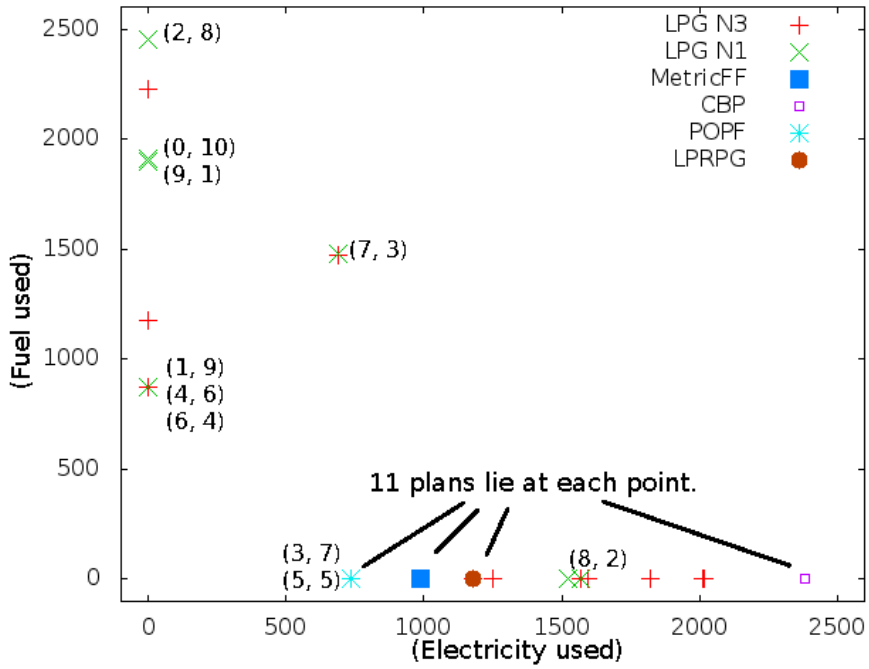


FIGURE 3.2: Results for 11 runs of each of the planners using 11 different weighted metric functions. Plans generated by LPG N1 contain additional annotation with the weights on the plan metrics under which the plans are generated.

In contrast, LPG did find multiple solutions. In fact, LPG N3 generated around 10 different solutions among the total of 30 solutions generated, because option -n 3 generates three plans each time it is invoked. Many plans overlap due to the small size of the problem, therefore, in the figure, less solutions is visible. For LPG N1 the solutions are annotated in Figure 3.2 with the weights on plan metrics which generated respective solutions. An interesting observation here is that solution with metric value (1523, 0), representing a plan which uses 1523 units of electricity and 0 units of fuel, was generated with the plan metric 8*(electricity) + 2*(fuel), meaning that a unit of electricity is 4 times more expensive than a unit of fuel. Similarly, the plan with coordinates (0, 1909) was generated using two weightings: (9, 1) and (0, 10). This demonstrates how unpredictable the solutions of LPG are.

To understand the mechanisms which help LPG generate a frontier of plans we have conducted further experiments where, for each of the 11 weights, we have run LPG multiple times. Figure

3.3 represents the results obtained by doing so. Here we can see that many of the plans generated violate Definition 3.1, that a metric sensitive planner should generate plans such that if a plan π_1 is generated under metric m_1, then there should not exist a plan π_2 generated under a different metric m_2 for which the evaluation using m_1 gives $m_1(\pi_1) \geqslant m_1(\pi_2)$.

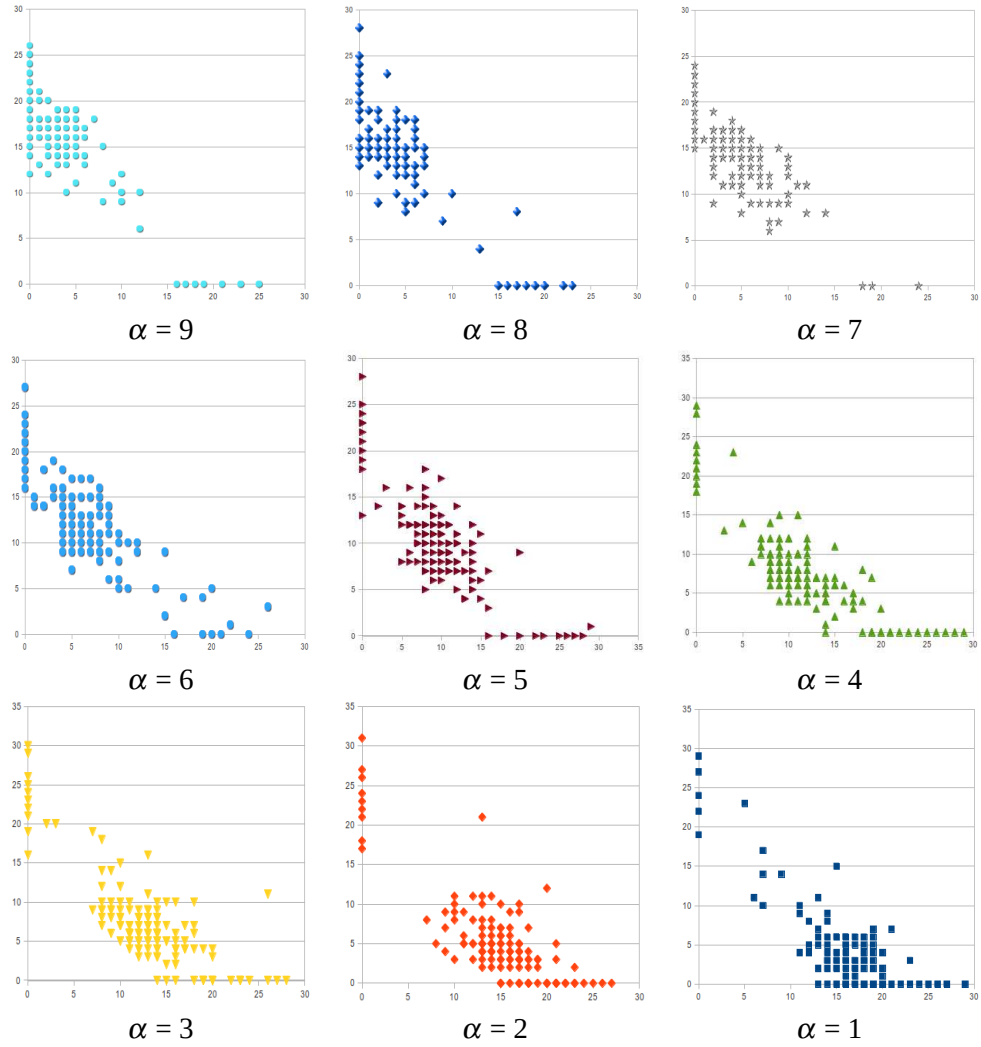


FIGURE 3.3: Representation of results for multiple runs of LPG on sets of objectives. Plan metric is constructed as: $\alpha * (\textit{electricity-used}) + (10 - \alpha) * (\textit{fuel-used})$.

However, since LPG is a stochastic planner we can use Definition 3.3. It can be seen that the centroids of the sets from Figure 3.3 meet the criterion.

3.4.3 Conclusion on current state-of-the-art

In this section we have presented a survey of current state-of-the-art planners including LPRPG, POPF2, CBP, LPG, MetricFF, and discussion of optimal planners. This discussion is presented in the context of domains with state dependent action cost. We have stated that the main limitation of many current optimal planners is the lack of support for metric fluents, and the fact that the action cost is limited to constant predefined values. Due to their speed, they can be only used on small instances of problems. This makes generating multiple plans, for the purpose of building sets of plans, impractical. The satisficing planners have all been described in more detail in Section 2.7. By a simple example we could see that only LPG behaved in a metric sensitive way. Unfortunately many current state-of-the-art planners use plan length as a proxy for plan quality. In many current benchmark domains, this approach is sufficient to generate good quality solutions and therefore achieve a sense of metric sensitivity. Some planners, like CBP, which use metrics to guide them in their search, appear metric insensitive due to their current inability to reason with state dependent action costs. We also assume that metric sensitive behaviour of LPG can be at least partly attributed to its stochasticity. We investigate this further in Section 3.7.

3.5 Related work

This section briefly presents work related to finding plans using non-constant action costs. It highlights the importance of domains containing this property as they become more widely used to solve real world problems.

An example of a domain where the ability to reason with a consideration of a plan metric matters is machine tool calibration [47, 48]. Machine tools are used to cut metal in manufacturing processes. As part of their maintenance, these tools need calibrating. In order to calibrate the tool, the precision of its components is measured. This process requires a calibration plan,

which is traditionally constructed by an engineer. The process of creating a calibration plan by hand is difficult, time consuming and increases the period required for maintenance. Automatic calibration plan generation, similarly to the manual method, aims at minimizing the time it takes to calibrate the machine and the accumulated uncertainty of the measurement. In their work, Parkinson et al. [48], used LPG to generate calibration plans which would minimize a plan metric made of the measurement uncertainty and the plan makespan, where the plan makespan is used as a proxy for the calibration time. Both metrics are weighted equally. However, the authors claim that this is not a constraint and, potentially, a user could alter these weights. This is consistent with findings in this thesis associated with the use of LPG.

Keyder and Geffner [34] present one approach to solving domains with monotonic plan metrics. First the method constructs the RPG for a given problem. Then, using calculated action costs, it constructs the cheapest plan achieving the goal within the RPG. This is done backward from the goal, taking the cheapest achiever for each open goal or precondition. An important observation here is that the action candidates for fact achievers are only the actions which appear in the RPG. The outcome of the heuristic is better than simply taking arbitrary achievers, and the approach favours cheaper plans. However, it is still prone to the bias demonstrated in the Example 3.1. The actions that do not appear in the RPG, such as driving via L1, are not considered in the construction of the relaxed plan. Therefore the method cannot find cheap solutions if there exists an expensive but short solution. The method does well on domains where the plan cost and plan length are strictly correlated, Definition 2.4.

An effort to overcome the limitations of RPG has been first visible in the work of Sapena and Onaindia [54]. They propose a modified RPG construction where the actions are added as in the standard RPG and action effects are postponed by the cost of the action achieving them. In order to calculate the cost by which effects of an action a are postponed a sum is taken of two components: the cost of achieving a, and the cost of applying a. Where the cost of achieving a is the sum of the costs of all of its preconditions. The new RPG prefers plans

TABLE 3.1: Summary of heuristic functions used in comparison between their quality and time [18].

Heuristic	Based on	Description
h_{add}	Classical RPG, Cost of goals	$\sum_{g \in Goal} cost(g)$
h_{max}	Classical RPG, Last RPG layer	$\sum_{p \in LastRPGLayer} cost(p)$
h_{mff}	Classical RPG, Relaxed plan	$\sum_{p \in RelaxedPlan} cost(p)$
h_{sin}	Cost RPG, Relaxed plan	$\sum_{p \in CostRelaxedPlan} cost(p)$
h_{level1}	Cost RPG, Relaxed plan	$\sum_{p \in CostRelaxedPlan} cost(p)$
h_{level2}	Cost RPG, Relaxed plan	$\sum_{p \in CostRelaxedPlan} cost(p)$

cheaper in terms of the cost rather than shorter plans. The authors mention a problem with the context-dependent action cost, but do not offer a solution.

A similar construction is explored by Fuentetaja et al. in the construction of a Cost-Based Planner (CBP) [19]. The difference between their approaches is that Fuentetaja et al. postpone the application of an action, whereas Sapena and Onaindia the insertion of action effects. Their planner is further described in Section 2.7.3. In their work Fuentetaja et al. [17, 18] examine the trade-off between the runtime and the quality of solutions. The experiment is based on two search algorithms, A^* and CEHC, where the latter is a version of Enforced Hill Climbing (EHC) adapted to handle cost more effectively. Five heuristic functions are used in the comparison. These functions are summarized in Table 3.1. Where h_{level1} and h_{level2} differ in how a $cost\_limit_i$ for RPG layers is computed. For h_{level1}, the cost of the next RPG layer is calculated as: $cost\_limit_i = min_a(cost(a) + cost\_limit_{i-1})$ for $a \in OpenAdd$, where $OpenAdd$ is the list of actions pending to be inserted. This means that the h_{level1} heuristic assumes the cost of an action to be the cost of applying it plus the cost of its preconditions as represented by the cost at the previous fact layer. The cost for the next action layer within the h_{level2} heuristic is calculated as $cost\_limit_i = min_a(cost(a) + cost\_limit_k)$, where the $cost\_limit_k$ is the cost of the first fact layer in which all of the preconditions of a appear. Based on their findings the h_{level1} heuristic gives the best quality estimate and takes a comparable amount of time.

The work of Sapena and Onaindia and Fuentetaja et al. deals effectively with domains with a

simple cost model. Although the main limitations of the state-of-the-art planners to constant action cost is addressed, there remains the limitation on the context-dependent action cost, which occurs in more realistic models. What is more, all of the domains, used in the evaluation of the work presented above, can be grounded, given the initial state description, to abstract the cost of actions from values of the metric fluents.

3.6 Simulating metric sensitivity

This section presents a method for generating diverse solutions using planners, which are not metric sensitive. The planners used in this experiment did not respond to the change of the metric, as demonstrated in Section 3.4.2.1. The focus of this experiment is to examine whether MetricFF or LPRPG can behave in a way which simulates metric sensitivity.

The approach is to impose bounds on plan metrics, including any of the lower or upper bounds on one or multiple objectives/resources at the same time. By limiting the value of a plan metric, it is forced to find alternative solutions. Therefore, the planners can be "pushed" to explore different areas of the search space. Because these bounds use metric fluents, if the behaviour of a planner changes, it means the planner has modified its behaviour in response to the metrics and this could be seen as metric sensitive behaviour. It is important to note that although we say that the planner becomes metric sensitive, it does not generate different solutions for different metrics without imposing the special bounds on the metrics.

It it obvious that imposing an upper bound on certain metrics causes the planner to search in different areas. In practice, this usually involves arriving into many dead-ends, which further demonstrates that planners do not reason well with metrics. We now try a less intuitive approach which generates fewer dead-ends and has less impact on the performance of the planners we consider. Instead of upper bounds on plan metrics, we impose lower bounds.

In order to demonstrate the impact of the lower bounds on the output of the planner for each considered plan metric we impose a lower bound by adding the following to the goal: $(>= (\text{plan-metric-name})\text{N})$ Where N is the bound. In this experiment we use lower bounds of minimum 0, 10, 20, 30, 40 and 50 units on fuel and electricity and all of their combinations, which gave us 36 different bounds. Starting with (0,0) meaning use at least 0 fuel and 0 electricity units, then (0, 10), (0, 20) until (50, 50) meaning use at least 50 units of fuel and 50 units of electricity.

In Figure 3.4 we present sets of results for planners which were unable to generate good quality sets in a weighted approach. We show a frontier of plans generated by LPG for the same problem instance for reference. MetricFF and LPRPG were given lower bounds on the plan metrics which forced them to use minimum amounts of each plan metrics and therefore exhibits the trade-off between these metric.

It is clear that this approach of adding bounds on plan metrics increased the variety of the results achieved by these planners. They are also comparable with the plans found by LPG in the approach where multiple objectives were present.

Although this approach could successfully generate sets of diverse plans, lack of control on where the plans were found does not allow for comparison with metric sensitive approaches.

3.7 The impact of stochasticity on the quality of plans

One way to attempt to generate plans that populate an APF is to generate plans non-deterministically and hope that the random sample of solutions includes plans that are of high quality with respect to *some* combination of the metrics. This approach can be taken in combination with a genuine sensitivity to the metric (producing plans that are intended to be better, but also might be better by chance), or ignoring the metric. LPG is an example of a stochastic planner, but it does not ignore the metric. An interesting question, to which we return in our experimental

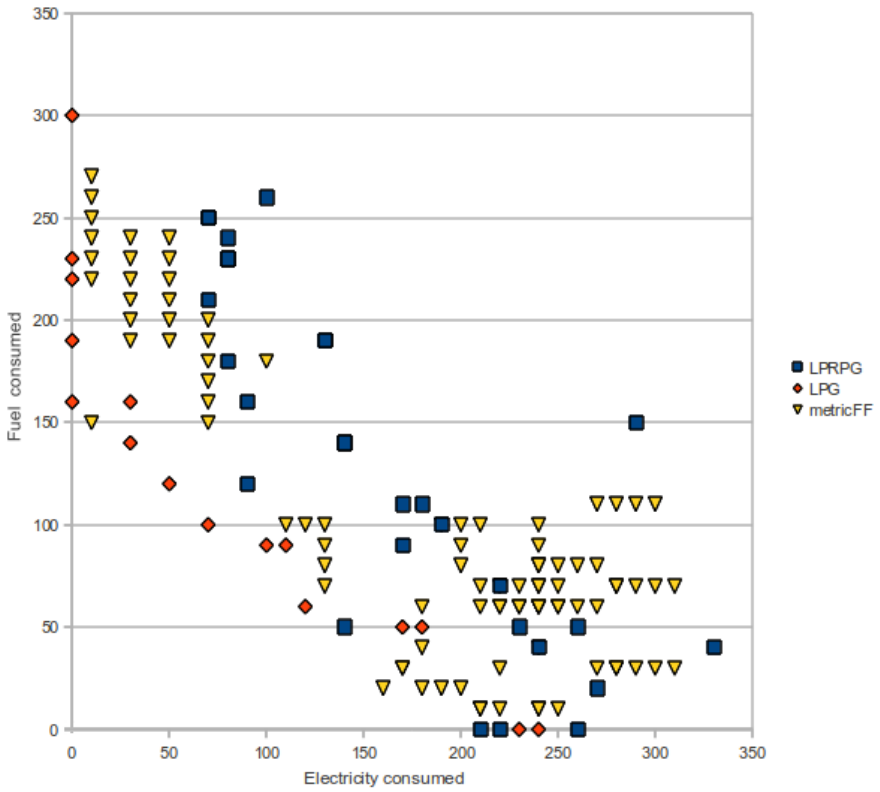


FIGURE 3.4: Results for runs of each of the planners using lower bounds on metrics as described in Section 3.6.

evaluation, is the extent to which LPG can populate an APF because of its efforts to optimise the plan metric and the extent to which it is a consequence of its stochastic behaviour leading to diversity in its solutions. Stochasticity does not provide significant benefits when dealing with a single metric problem, but when populating an APF within the framework described in Section 5.6, the collection of *all* plans produced is kept and searched for non-dominated solutions. Therefore it is possible, that a plan produced for one weighting scheme might be the best solution under a *different* weighting scheme.

In Section 5.7.3, on the generation of frontiers of plans, it is shown that adding stochasticity to a planner significantly improves APF generation in terms of quality and distribution of the frontier. In this chapter we aim to measure whether this improvement in APF quality comes

TABLE 3.2: Plan Metric using different metric fluents depending on a domain.

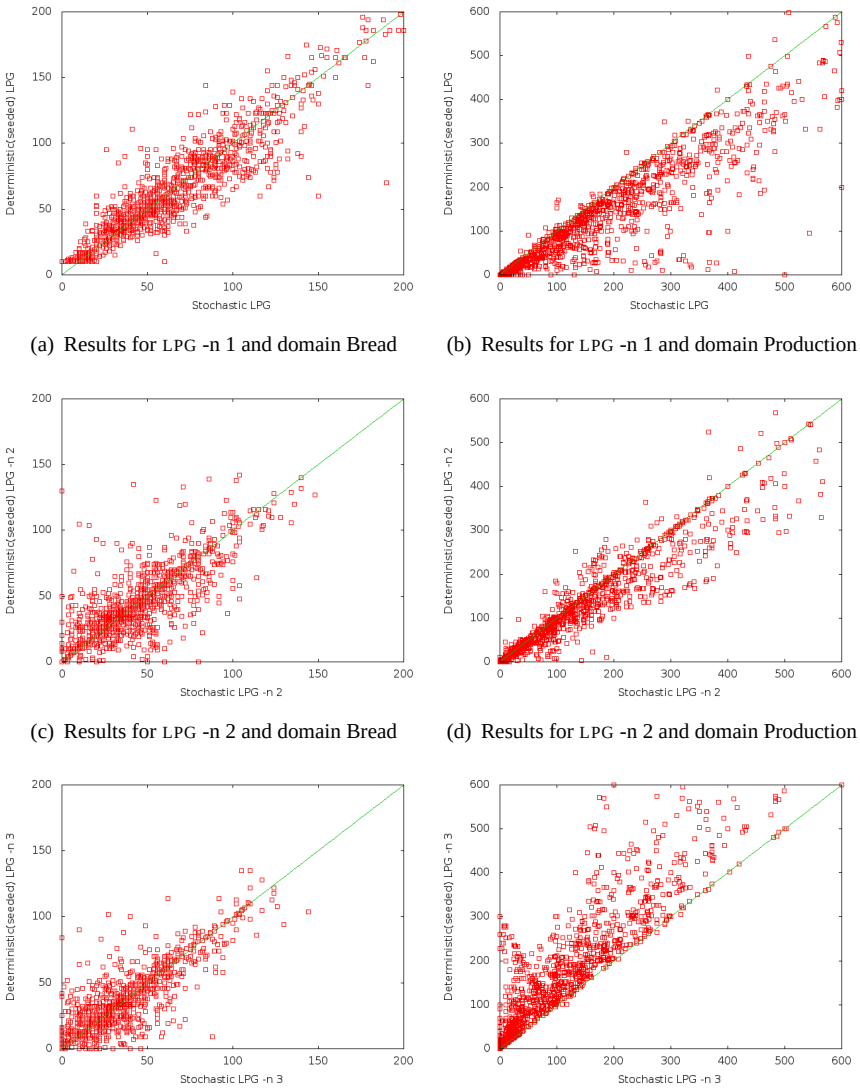
Domain	Plan Metric
Bread	A*(energy)+B*(labor)+C*(pollution)
Production	A*(labor)+B*(hazard)+C*(machine-cost)
Driverlog	A*(time-walked)+B*(fuel-used)+C*(electricity-used)
Driverlog Metric	A*(time-walked)+B*(fuel-used)+C*(electricity-used)

from increased metric sensitivity of stochastic planners, or maybe their other properties.

3.7.1 Impact of stochasticity on performance of LPG

As mentioned LPG is a stochastic planner, and from the previous experiments, it seems like stochastic behaviour does help it in generating good solutions. In order to test how LPG would perform as a deterministic planner, we fake the determinism by seeding the random search with a constant value for every run. That gives consistently the same results every time the planner is run. Although it appears to be a deterministic planner what we effectively do is take a snapshot of a stochastic planner. This means that we cannot claim it to be fully deterministic even though it always outputs the same results for the same input. By using different value as the seed it is possible to manipulate the quality of the plans generated by LPG. Despite this ambiguity, whether LPG is or is not acting as a deterministic planner, we believe it gives a good indication on the impact on results when compared with the truly stochastic version re-run multiple times.

To check the difference in the generation of plans with and without stochasticity, a planner is run in its stochastic and deterministic version. The experiment is carried out for domains Bread, Production, Driverlog and Driverlog Metric, as described in Section 2.5. For each of the problems from each of the domains a set of 66 plan metric functions is generated, equivalent to the 66 solutions of a simplex A+B+C=10, for A,B,C all integers. The plan metrics for each weighting, depending on the domain are presented in Table 3.2.



(a) Results for LPG -n 1 and domain Bread

(b) Results for LPG -n 1 and domain Production

(c) Results for LPG -n 2 and domain Bread

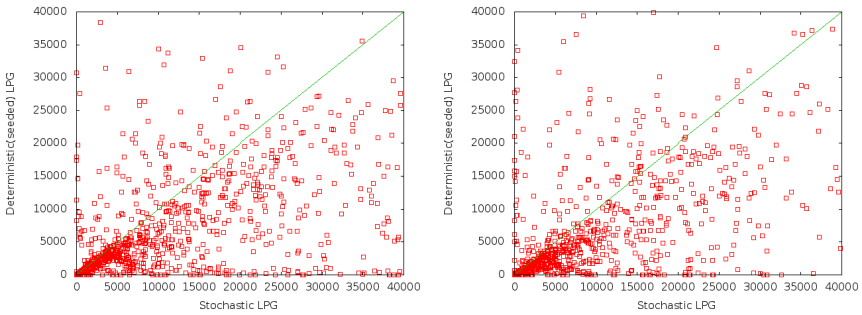
(d) Results for LPG -n 2 and domain Production

(e) Results for LPG -n 3 and domain Bread

(f) Results for LPG -n 3 and domain Production

FIGURE 3.5: Results for all weights and all problems and weightings aggregated by domain. Images a), c), e) represent comparison of stochastic and seeded version of LPG -n 1, -n 2 and -n 3 respectively on domain Bread. Images b), d) and f) represent the same comparison for Production domain.

Deterministic planners are run once, and stochastic planners are run ten times for each weighting, for each problem, for each domain. The resulting quality of a solution is then compared on a graph where the X axis represents the values for stochastic configuration and the Y axis



(a) Results for LPG -n 1 and domain Driverlog

(b) Results for LPG -n 1 and domain Driverlog metric

(c) Results for LPG -n 2 and domain Driverlog

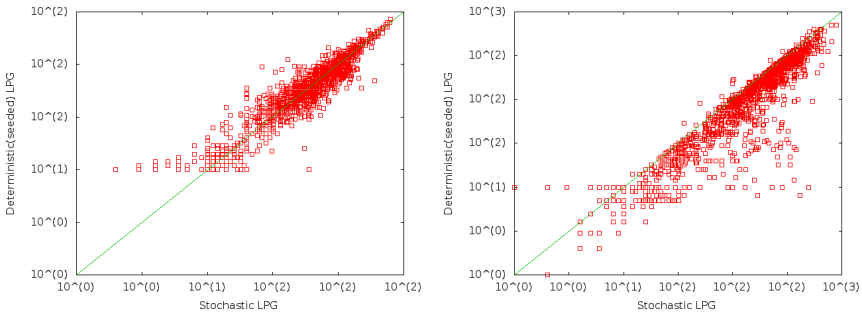
(d) Results for LPG -n 2 and domain Driverlog metric

(e) Results for LPG -n 3 and domain Driverlog

(f) Results for LPG -n 3 and domain Driverlog metric

FIGURE 3.6: Results for all weights and all problems and weighting aggregated by domain. Images a), c), e) represent comparison of stochastic and seeded version of LPG -n 1, -n 2 and -n 3 respectively on domain Driverlog. Images b), d) and f) represent the same comparison for Driverlog metric domain.

for deterministic. Each image contains an X=Y line to separate the areas where the deterministic configuration generates better solutions, from the areas where the stochastic configuration



(a) Results for LPG -n 1 and domain Bread

(b) Results for LPG -n 1 and domain Production

(c) Results for LPG -n 2 and domain Bread

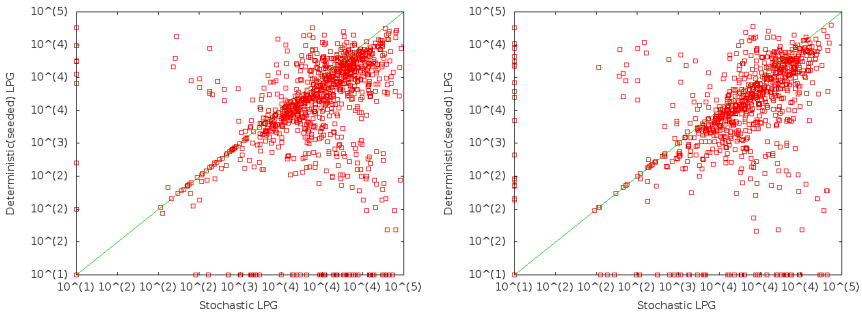
(d) Results for LPG -n 2 and domain Production

(e) Results for LPG -n 3 and domain Bread

(f) Results for LPG -n 3 and domain Production

FIGURE 3.7: Logarithmic scale results for all weights and all problems and weightings aggregated by domain. Images a), c), e) represent comparison of stochastic and seeded version of LPG -n 1, -n 2 and -n 3 respectively on domain Bread. Images b), d) and f) represent the same comparison for Production domain.

generates better solutions which coincides with below and above the X=Y line respectively. Therefore if more plans are shown below the X=Y line, then more of the plans generated by the deterministic configuration for the same weights have better quality.



(a) Results for LPG -n 1 and domain Driverlog

(b) Results for LPG -n 1 and domain Driverlog metric

(c) Results for LPG -n 2 and domain Driverlog

(d) Results for LPG -n 2 and domain Driverlog metric

(e) Results for LPG -n 3 and domain Driverlog

(f) Results for LPG -n 3 and domain Driverlog metric

FIGURE 3.8: Logarithmic scale results for all weights and all problems and weightings aggregated by domain. Images a), c), e) represent comparison of stochastic and seeded version of LPG -n 1, -n 2 and -n 3 respectively on domain Driverlog. Images b), d) and f) represent the same comparison for Driverlog metric domain.

The results for these experiments are presented in Figures 3.5 on page 52 and 3.6 on page 53 using a linear scale and in Figures 3.7 on page 54 and 3.8 on page 55 using a logarithmic scale

to focus on different areas of the solution space.

An interesting observation is the change from the deterministic version of LPG performing better, to the stochastic version of LPG performing better which can be observed as the optimisation option changes from -n 1 to -n 3. This is visible for domains Production Figure 3.5 b), d), f) on page 52 and Figure 3.7 b), d), f) on page 54 and both Driverlog domains presented in Figures 3.6 on page 53 and 3.8 on page 55. Although this property is not clearly visible for the Bread domain in Figure 3.5 a), c), e) on page 52 and Figure 3.7 a), c), e) on page 54, it shows that the stochastic decisions in search help it to find solutions of high quality which are not found when run with a seed, when used in configuration with -n 3 option.

Aggregated numerical results for this section are presented in Appendix C.

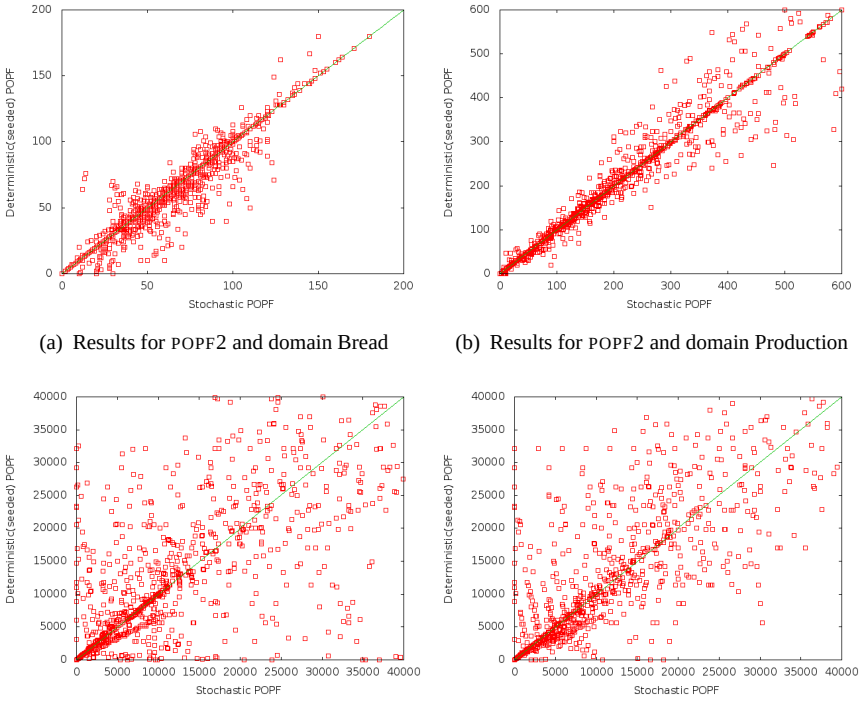
3.7.2 Stochastic POPF2

This section describes a second experiment which aims at measuring the impact of stochasticity on the metric sensitivity of POPF2. The purpose of this experiment is to show how the metric sensitivity of the planner is influenced by uninformed stochasticity. Two versions of POPF2 are used; original and with added stochasticity.

The changes to POPF2 were minimal and only included adding stochasticity in two places. First, in heuristic evaluation, where every heuristic value is incremented or decremented by a small random number. The second place where stochasticity is introduced is in branch ordering: the order in which actions are considered when expanding a state from which heuristic gives no clear guidance. As opposed to LPG, POPF2 is a deterministic planner and the stochasticity which is added is not integrated with its search in a clever way. LPG uses stochasticity in an informed way in the sense that it helps it in discovering areas of the search space that would not normally be discovered, but might also be helpful.

Similarly to the previous experiment to check the difference in generation of plans with and without stochasticity, a planner is run in its stochastic and deterministic version. The experiment is conducted in the same way as before using 66 weightings for all problems and domains. The plan metrics for each weighting, depending on the domain are presented in Table 3.2 on page 51.

The results are present in Figure 3.9 on page 57 using a linear scale and in Figure 4.9 on page 88 using a logarithmic scale to focus on different areas of the solution space.



(a) Results for POPF2 and domain Bread

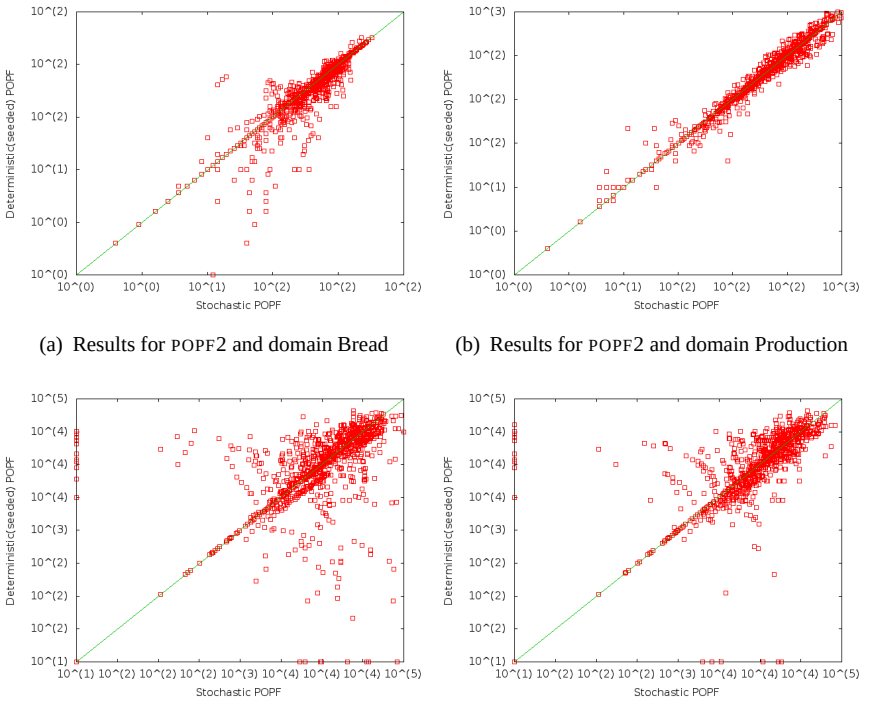
(b) Results for POPF2 and domain Production

(c) Results for POPF2 and domain Driverlog

(d) Results for domain POPF2 and Driverlog metric

FIGURE 3.9: Results for all weights and all problems and weightings aggregated by domain. Images represent comparison of stochastic and deterministic version of POPF2 on domains Bread, Production, Driverlog and Driverlog metric pictured on images a), b), c) and d) respectively.

The opposite behaviour to the one exhibited by LPG, when stochasticity is introduced and removed, can be seen for POPF2 presented in Figures 3.9 on page 57 and 3.10 on page 58.



(a) Results for POPF2 and domain Bread

(b) Results for POPF2 and domain Production

(c) Results for POPF2 and domain Driverlog

(d) Results for POPF2 and domain Driverlog metric

FIGURE 3.10: Logarithmic scale results for all weights and all problems and weightings aggregated by domain. Images represent comparison of stochastic and deterministic version of POPF2 on domains Bread, Production, Driverlog and Driverlog metric pictured on images a), b), c) and d) respectively.

A possible explanation is the difference between the way stochasticity is introduced into POPF2 and LPG. In LPG stochasticity is an integral part of the search algorithm which helps it to avoid getting stuck in dead ends and helps in exploring more areas of the search space. The stochasticity in POPF2 only reorders the way in which it explores states slightly and is completely uninformed. This is reflected in the results where we can see that it both improves and deteriorates the quality of a solution in a stochastic way.

Although the quality of solutions is not significantly affected by the introduction of stochasticity, POPF2 with stochasticity could generate a larger variety of solutions which we show, in Section 5.5, is a very useful for the APF generation.

3.8 Conclusion

In this chapter we define metric sensitivity and discuss the current state-of-the-art planning technologies in terms of exhibiting metric sensitive behaviour. The chapter starts with an example which highlights the importance of using metric fluents for plan evaluation. We describe planners which can handle domains where plan quality is defined by cost of actions and not plan length. Context-dependent action cost is a serious limitation of the current state-of-the-art planners. These planners, in most cases, did not exhibit metric sensitive behaviour when faced with a problem containing context-dependent action cost, which had to be calculated from metric fluents. An evaluation of their performance is presented in section 3.4.2.1 and the best candidate, LPG is selected for further comparison.

The importance of solving problems containing context-dependent action cost is currently becoming more visible in various research areas. These areas, along with the related research, are discussed in Section 3.5. Solutions to many real life problems requires the use of more sophisticated action cost schemas.

The main contributions from this chapter are:

- Definition of metric sensitivity.
- Exploration of the impact of stochasticity on metric sensitivity of planners.

Chapter 4

Achieving metric sensitivity

The need for control over where, in the metric space, the planner generates solutions is among the reasons for developing a metric sensitive planner. In the previous chapter (Chapter 3), it is shown that, among the state-of-the-art satisficing planners, the only planners which exhibit metric sensitive behaviour, in domains with state dependent action cost, are stochastic. The coupling between the stochasticity and metric sensitivity impacts on the precision of the planner.

In this chapter we present a novel method for obtaining a deterministic metric sensitivity. This method is based on a cost-based Relaxed Planning Graph (cRPG) [57] and can effectively reason with context-dependent action costs. An implementation of the cRPG via translation to temporal domains and evaluation of this method in comparison with the metric sensitive state-of-the-art concludes this chapter. The evaluation of the method contains a discussion on the impact of adding stochasticity to the new compilation method.

4.1 New cost-based RPG as an approach for metric sensitivity

In order to overcome the limitations of most current planners in reasoning with context-dependent metrics, we propose a new approach to calculating a heuristic that favours cheaper plans in terms of defined metric cost instead of plan makespan. Following the description of the RPG, presented in Section 2.6, we propose an extension to the RPG to enhance its metric sensitivity. We conclude with a discussion of its capabilities and limitations.

4.1.1 Pitfall of RPG

Consider domain represented in Figure 2.3 and the RPG constructed for this domain as shown in Figure 2.4. As can be seen from this example, the RPG approach is biased towards shorter plans in terms of the number of actions, or layers in the RPG. Current RPG based planners which handle metric fluents, are also prone to this bias because, although they keep metric values, they still expand the RPG according to the same algorithm. This means that at the stage where the goal is satisfied by a layer in RPG, the expansion stops. However, there still might be a better plan, in a metric sense, which cannot be extracted from the RPG. In order to find this plan the RPG must be expanded further.

4.1.2 The cost-based RPG

In this section we introduce a new idea of obtaining a metric sensitive heuristic using a modified version of the RPG. The modification of RPG is similar to the one introduced by Sapena and Onaindia [54] and Fuentetaja et al. [19]. Where the two approaches differs in postponing the insertion of actions or action effects into RPG. We propose an extension where to each of the fact layers in the RPG a cost label is added. These labels are used to denote the cheapest cost of achieving each fact layer, similar to h_{max} heuristic, for all facts in the layer. Assigning cost labels to different layers of the RPG is similar to the approaches taken by Sapen [54] and Fuentetaja [19], which is also similar to the TRPG in POPF which uses time as the label. It

allows the new heuristic, generated using the new cost-based RPG, to be more metric oriented. In contrary to the work of Sapena and Onaindia, and Fuentetaja et al. we split each action into the start and end of an action. A more detailed description of the algorithm including the handling of context-dependent action cost is given below.

We start by describing the cRPG, an example of the cRPG and later focus on the method in more details by describing how context-dependent action cost is handled.

4.1.2.1 General idea

The key difference between the cRPG and the RPG is the additional information, associated with the cost, carried with every fact layer of the graph. Each fact layer is now labelled with the cost of achieving the facts in this fact layer. When constructing the graph we make sure that we only apply the cheapest actions that achieve the facts. We can now determine what is the minimum cost of achieving each of the facts within cRPG.

The algorithm for construction of the cRPG can be summarized as:

1. Split all actions into start and end action such that:

   The start-action contains all preconditions of the action and its disabling metric effects.

   The end-action contains all enabling effects of the actions

2. Construct the first fact layer of the cRPG using facts and metric fluent values from the initial state, and assign it cost label of 0.

3. Add all starts of actions applicable in the state described by the previous fact layer. Push all ends of these actions on a priority queue based on the action cost.

4. Create next fact layer by merging the previous fact layer with all effects of the start actions applied in the previous step.

5. Create a new action layer by popping the cheapest action ends from the priority queue.

6. Create a new fact layer using the effects of end actions from the previous step and all facts from the previous fact layer. This fact layer is labelled with the cost equal to the cost of a layer when the start actions were applied plus the cost of the actions applied.

7. Repeat from step 3.

The construction method of the cRPG, the method for selecting which action to apply and the process of extracting the relaxed plan, is similar to the TRPG used in POPF. Each action is split into a start and an end action. The start of the action becomes applicable whenever all preconditions for the original actions appear in the preceding fact layer. The end of an action is applicable when the start of the action was applied and the cost of the current fact layer is distant, from when the start action was applied, by the cost of applying the action.

There are differences between these two approaches. For example, the end action in the TRPG is applicable after a certain time elapses from when the start of the action was applied and its end preconditions are met. In the cRPG the end action is always applicable after a certain cost is incurred, and it does not depend on any additional conditions. That makes scheduling of actions much simpler as the end actions do not need to move in the metric space.

4.1.2.2 Example cRPG

In this section we present an example cRPG constructed for the problem described in Figure 2.3. For the sake of the example assume the metric function to be:

```
(:metric minimize
    (+
      (* 1 (electricity-used))
      (+
```

```
        (* 1 (fuel-used))
        (* 8 (walked))
      )
    )
)
```

which means that each action that uses 1 unit of electricity or fuel increases the metric function by 1; also if an action requires a person to walk it increases the metric function by 8 units. Therefore using fuel or electricity is more desirable within this weighting.

Figure 4.1 represents an extract of the cost-based RPG. Action costs for the cRPG as in Figure 4.1, for the purpose of illustrating the cRPG construction, are all assumed to be at least one. Using the above metric function the drive action has cost equal to the distance driven, as depicted by Figure 2.3, multiplied by one (the usage of fuel or electricity).

As opposed to the RPG (Figure 2.4 on Page 2.4), the cRPG (Figure 4.1 on Page 4.1) recognises that applying drive action from L0 to L2, after fact layer with cost label 1, consumes much more resource (in terms of the current metric) and postpones its completion. The plan in which trucks drive through locations L1, L3, L4, L5 is obviously cheaper in terms of resource consumption and is preferred under this metric.

4.1.2.3 Handling context-dependent action costs

Handling context-dependent action costs is a very important aspect of this work. As we have seen in the Example 3.1, action costs can depend on the context in which they are applied. In the case of the aforementioned example, it was the load of the truck. We do not know how much a drive action is going to cost until the time of application, when it is possible to check the load of the truck being driven.

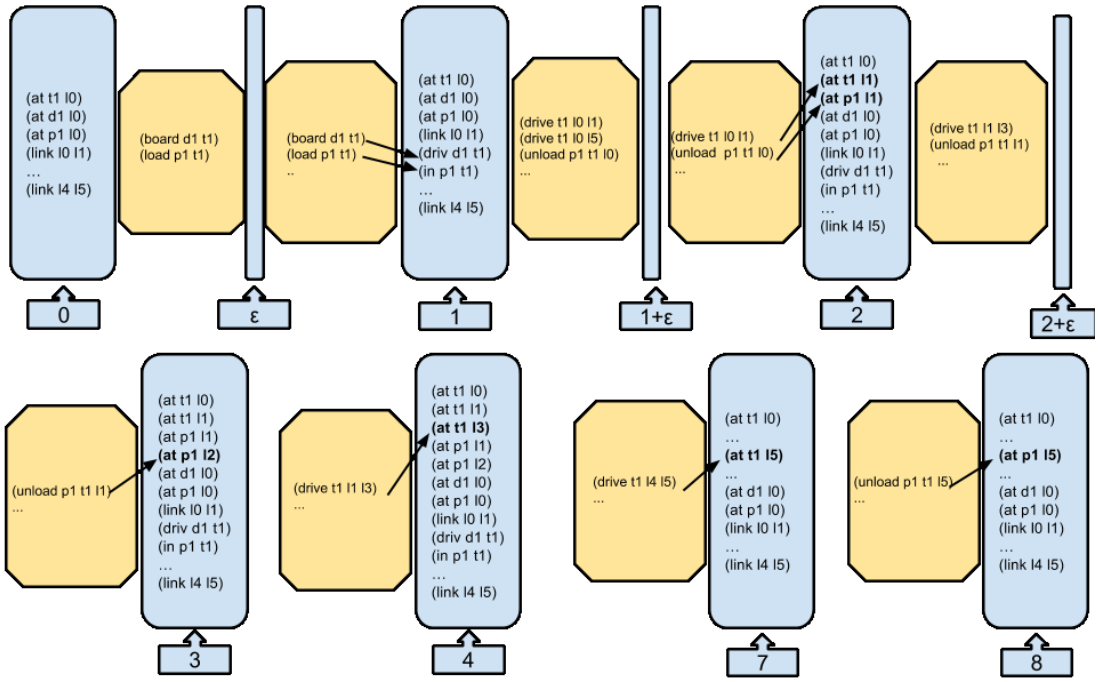


FIGURE 4.1: Diagram represents extract of the cRPG build for problem represented by Figure 2.3 from Page 27

While building the cRPG, the action cost is calculated in every state, just before the cRPG construction, and used through the construction of the cRPG. It is possible that, while constructing the cRPG, the actual cost of the action varies between action layers (in the previous example, that would happen if previous cRPG layers contain a load action). In this case we continue to use the pre-calculated cost as an estimate in that given state. This means that in the further layers of the cRPG, context-dependent action costs are prone to error. At the same time, with the progression of the state in search, this error is alleviated. This is caused by the more accurate estimation of metric fluents at the start of the cRPG creation in a state. Therefore by advancing in the search space we eventually eliminate the error in the cost.

This behaviour causes some limitations which are further discussed in Section 4.1.2.7.

4.1.2.4 Metric fluents in the cRPG

Another important aspect of reasoning with numbers is the handling of values of metric fluents in the RPG. The most common approach is to represent each metric fluent using upper and lower bounds. Every time the fluent is increased, the upper bound is updated. Every time the fluent is decreased the lower bound is updated. The issue with this approach is that the bounds diverge very quickly. A reasonable, possible value of a metric fluent is then difficult to determine. An action becomes applicable if any value between the bounds on the metric fluent satisfies its preconditions.

Consider the RPG constructed in Figure 2.4, and fuel used by an action given as (fuel-used) = (distance ?L1 ?L2)*(load ?t). After action layer 2, where two drive actions take place (drive t1 L0 L1) and (drive t1 L0 L2), the upper bound on (fuel-used) is increased by both actions. This creates the following bounds on the value of (fuel-used) $\in [0..11]$, because (load ?t) is assumed to be equal to 1 at start. The bounds on (load ?t) are $[1..2]$, however having more packages at location L0 would cause this load to grow significantly.

This large growth of bounds on metric fluents is one of the reasons for handling context-dependent action costs before the cRPG construction starts.

4.1.2.5 Incrementing metrics

One important aspect of splitting the action into two and constructing the cRPG is how metric change is treated. If an increment/decrement of a metric function appears in the effects of the original action, when constructing the start and end action we need to make sure that the enabling effects are applied with the end-action and disabling changes with the start-action. By enabling changes to the metrics we do not mean positive changes, but a change which could make other actions applicable. For example incrementing an amount of dough in the Bread domain [50] enables action for forming buns. At the same time decreasing the amount of load

of a truck, when full, enables more load-truck actions to be applied. We can detect when incrementing or decrementing should be treated as en enabling effect by searching preconditions of actions for expressions involving the metric function being incremented/decremented. Similarly for disabling effects, which do not mean decreasing the value of a metrics but making actions not applicable. This situation occurs when an action uses up resources needed by other actions. This situation can also be detected by checking preconditions of actions for expressions such as (< (metric) #VALUE).

A more complex situation arises when one metric appears with a enabling and disabling effect. In other words, some actions require smaller and some actions require larger values of the metric. In that case, for completeness, the algorithm replaces this metric with two new metrics $metric^+$ and $metric^-$ which are the metrics with enabling and disabling effects respectively, and use them throughout the problem description.

Every time the metric was increased or decreased we increment or decrement both $metric^+$ and $metric^-$ by the same value. Although both metrics hold the same value, they are incremented and decremented at a different stage of action execution. This keeps the problem consistent with the original one. The enabling $metric^+$ metric is altered always at the end of an action, and the $metric^-$ is altered at the beginning of an action. Every occurrence of the metric in an enabling precondition is replaced with $metric^+$. Similarly, every occurrence of the metric in a disabling precondition is replaced with $metric^-$.

4.1.2.6 Heuristic extraction

When the cRPG is built and the last layer of the cRPG satisfies the goal condition, a heuristic value can be extracted from it. There are multiple ways in which a heuristic function can be extracted. We first describe two methods of extracting heuristic values and then focus on what information is carried by each of them.

The first, and simplest, choice of value to use as a heuristic function is the cost label of the last cRPG layer, similar to h_{max} heuristic. Since the cRPG was built with a monotonic cost increase, the last layer has a cost label "close" to the cost of the plan. This method does not provide information about the costs incurred by actions executed in parallel. For example, for two different trucks starting and finishing drive actions at the same layers, and each of them contributing Δ_{cost} to the overall cost, the effects of both actions are available after Δ_{cost} cost. However, the actual resource consumption, and therefore the actual cost of applying these two actions, is $2 * \Delta_{cost}$. Hence the second method for extracting heuristic value as the sum of all costs of actions from the relaxed plan from the cRPG is considered. This is similar to the h_{add} heuristic. The relaxed plan is extracted by starting from the layer containing the goal, and selecting the earliest achievers for all facts.

Consider a case where we have two identical trucks, t1 and t2, in location L0, two packages p1 and p2 which have to be delivered to L2 and L5 respectively and the network of connections between locations as in Figure 2.3 on Page 27. For the sake of argument, consider a simplified Driverlog domain where the fuel consumption depends only on the distance between cities, and not on the load of the trucks. The following two relaxed plans solve this problem:

- Load p1 onto truck t1, load p2 onto t2. Drive t1 to L2 and t2 to L5 via L3. Unload packages.
- Load p1 and p2 onto truck t1, Drive t1 to L5 via L3. Unload Package p2. Drive t1 to L2. Unload p1.

For the metric considered (fuel-used), the first plan is found at the cost layer 10 of the cRPG with cost of relaxed plan equal to 16. The second plan would be found on the cost layer 16 and the cost of relaxed plan is 16 as well.

This simple example illustrates that both values, the cost label of the layer containing the goal state and the relaxed plan cost can be treated as bounds on the actual cost.

The algorithm of finding the more accurate value involves the following steps:

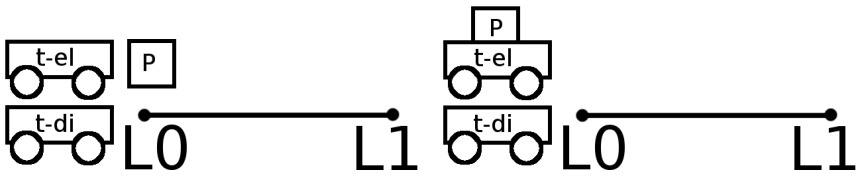
- Expand the cRPG until a goal is reached.
- Extract the cost of the relaxed plan.
- Expand the cRPG until the layer with cost equal to the previously extracted relaxed plan cost.
- Use the cheapest relaxed plan from the expanded cRPG.

This algorithm helps us get more accurate heuristic evaluation and prevents us from falling into the problems of the initial approach.

4.1.2.7 Interesting cases and limitations of the cRPG

Some limitations and interesting cases of the use of the cRPG heuristic are discussed in this Section. Methods to overcome them are presented in the discussion.

Cyclic context-dependent cost switch Consider a simple Driverlog example with two trucks, diesel and electric, one driver and one package. A diagram illustrating this domain is present in Figure 4.2 a). For the purpose of this example, the cost of electricity and fuel is equal to 10, the distance between the two locations is 10. The cost of driving a truck depends on the weight of the truck, its resource consumption speed, the distance and the cost of the resource. Where the cost of the resource is determined by the weight on a plan metric containing this resource. The weight of the truck is the sum of its initial weight and its load. Each truck has an initial weight of 10 units and consumes fuel in the speed of 10 units. The formula to calculate the total resource consumption is: $\textit{total-consumption} = \textit{weight} * \textit{distance} * \textit{diesel-consumption} + \textit{weight} * \textit{distance} * \textit{electricity-consumption}$. For the initial set-up, driving the electric truck from location L0 to L1, under a metric: **10 * total-consumption**, evaluates to 10 * 10 * 10 * 10 = 10000. The cost of driving the diesel truck is the same.



(a) Initial situation

(b) Situation when electric car is loaded with a package.

FIGURE 4.2: Diagram representing a simple Driverlog problem with diesel and electric truck.

At that stage the planner can select either of the actions as there is no difference between driving either of the trucks. Consider the case when the planner decides to use the electric truck. The first action in the plan is to load the package on to the truck; the state describing this situation is pictured in Figure 4.2 b). In this state the weight of the electric truck is increased by 1, to 11 units, to account for the package. Now the costs of driving to the destination using the electric vehicle is 11000 while using the diesel vehicle the cost is lowered to 10000. At this stage the planner decides to use the cheaper, diesel, vehicle and heuristic function produces (unload-truck p1 t-el L0) as a helpful action. This takes the planner back to the initial state, from which, when the diesel truck is loaded, a mirror situation occurs.

This problem can be overcome using states memoisation in search. When an action leads to the already visited state, it is not applied, and another best action is selected. If no helpful actions lead to a state which has not been visited, all actions are considered.

In the above example, this solution would try driving a loaded truck to the goal location, L1. This would allow the planner to notice that unloading the package achieves the goal and is therefore a good plan.

Context-dependent cost pitfall This issue is related to the context-dependent cost switch. Consider the Driverlog domain with electric and diesel trucks. For the sake of this example, consider the weighting under which electricity is much more expensive than diesel and therefore driving diesel trucks is more desirable. In this example, we also consider drive actions

where the amount of resources consumed depends partly on the load of the truck: load actions cause the drive actions to be more expensive. When evaluating a heuristic value from a state in which both trucks are empty, the method gives a relaxed plan using the diesel vehicle. This is a desired behaviour. Now, back in search, action (load ?p ?t_diesel) increases the load of the diesel truck, and therefore, when evaluating the state resulting from applying this action, the heuristic value is much higher than in a state resulting from applying (load ?p ?t_electric). The cheapest plan is still going to use a diesel vehicle. However, the load taken into consideration when calculating action costs is lower when the electric vehicle is loaded, as it does not appear in the relaxed plan, and therefore in the evaluation. This issue is overcome by using helpful actions. A pruning strategy is used in which only helpful actions are considered. This allows a much more focused exploration of the search space, and limits the branching factor of the search.

4.1.2.8 The cRPG construction summary

In this section we have presented an approach to building a cost-based RPG. This approach allows the planner to reason more effectively with plan metrics. Some of the limitations of the approach are also discussed with methods of overcoming them. We also give a method of reasoning with context-dependent action costs, using valuation of metric fluents in each state.

4.1.3 A comparison of the cRPG and the TRPG

The main difference between the cRPG and the TRPG is in the meaning and properties of their RPG layer labels. In the TRPG, parallel use of time is desirable and the total time taken by parallel actions is the maximum of the duration of the two. For the cost RPG, in contrast, using two actions in parallel consumes the sum of the costs incurred by both actions. This is an important difference, especially for the compilation based implementation of the cRPG, described in the next section.

A second important difference is when we satisfy all goal conditions in a certain layer of the RPG. In the cRPG the cost label gives a lower bound on the cost of the heuristic (and the relaxed plan), whereas in the TRPG the time label is equivalent to the duration of the relaxed plan.

There are many similarities in the way we construct both graphs. When building a TRPG an external check must be made to ensure that the time points of applying actions are feasible in POPF2. This is done using Simple Temporal Networks [12]. With regards to the cost, it needs to ensure that it is only possible to apply the enabling effects of an action if the cost of the action has been paid. This is similar to the duration of an action elapsed. In the cRPG, this is achieved by adding all enabling effects at the end and disabling metric effects at the start as described in Section 4.1.2.5.

Both the TRPG and the cRPG must deal with the problem of separating two start actions where one start of an action can enable the other. In the TRPG, it is achieved by separating both actions by a small time ε and we follow a similar strategy in the cRPG. If the duration/cost of both actions is equal, then the TRPG/cRPG must also separate their end by ε.

4.1.4 cRPG summary

This section presents a new approach to constructing a relaxed planning graph. The new construction allows for calculation of a more metric sensitive heuristic value. The cRPG construction is based on a well known, and very successful RPG. The modification of RPG is similar to the one introduced by Sapena and Onaindia [54] and Fuentetaja et al. [19]. It is also similar to the TRPG construction and the differences between both are further discussed in Section 4.1.3. A discussion of limitations and methods to deal with them is present in Section 4.1.2.7.

A method for extracting a relaxed plan from the new structure is presented and discussed in detail. There are multiple ways in which this can be achieved and some benefits and limitations of each are discussed.

In addition to the heuristic, a method of handling context-dependent action costs, by calculating values for all metric fluents before each cRPG construction, is presented. Support for context-dependent action costs can be easily integrated with the cost-based RPG. This allows the cRPG approach to reason with complex metrics.

4.2 Compilation based implementation of the cRPG

In this section we focus on methods of implementing the cRPG. A novel compilation from metric to temporal domains is proposed. It is presented as a mean of exploiting the current temporal planners for implementation of the cRPG heuristic algorithm.

4.2.1 Compilation to temporal domain

This compilation exploits the similarity of the cRPG and the TRPG. Due to this similarity if we could translate the action costs into their durations, the TRPG built by POPF2 would therefore be very similar to the cRPG described in previous section.

Due to some differences between the cRPG and the TRPG, as described in Section 4.1.3, special care is required in formulating durative actions. Below we describe two cases where special care is required. The first is when the values of metric fluents change. The second is when calculating the heuristic. This is due to the fact that, when used in parallel, cost is an additive metric whereas time is not. We will now briefly discuss these two cases.

As mentioned in Section 4.1.2.5 positive metric changes should take place at the end, and negative at the start, of each action. If this is violated, the cRPG might add unnecessary actions too early (positive changes *enable* other action starts) which can result in expansion of unnecessary states in search. This is avoided by grouping functions into positively and negatively affected metrics where positive and negative are interpreted according to whether they enable or disable other actions from being applied. When translating our domain from metric to temporal we then need to take care of which metrics are incremented/decremented at action starts

and which at action ends. The compilation proposed here addresses these issues by checking preconditions of all actions and determining whether an increase of a metric is used to enable an action, disable an action or enable some and disable other actions from being applicable. The two former cases are relatively easy to handle as we only need to remember to apply the increase or decrease at the start for disabling (negative) changes and at the end for enabling (positive) changes of the metrics. The later case, where a change of a metric can be used to enable some and disable other actions is more complex, and we deal with it by splitting the metric into its positive and negative versions, then applying them as described in Section 4.1.2.5.

One additional detail to consider is the applicability of an end of an action. We have already noted that, after an appropriate cost is paid, the end action is always applicable in the metric based RPG. This is not always the case for the temporal RPG, after a certain amount of time elapsed, equivalent to the cost. When an end action is not applicable in the TRPG, although its duration passed, the action is rescheduled to start or end at a different time point. Therefore in the compilation we need to ensure that there are no end-preconditions that would make the end action not applicable. This is achieved by always setting all conditions at the start of an action, the end-action preconditions are then left empty.

4.2.1.1 Example action compilation

Based on the complexity of the translation, we divide the translation into two main categories: context-dependent and context-independent. Where a context-independent category also includes domains where the cost of an action is expressed as a function of a metric fluent, and this metric fluent is constant for all possible state transitions. We now present an example translation of both cases.

Consider the simplified action which can be translated to context-independent costs:

```
(:action DRIVE-ELECTRICTRUCK
:parameters (?v - electrictruck ?f ?t - location ?d - driver)
```

```
:precondition (and (at ?v ?f)  (driving ?d ?v)(link ?f ?t))
:effect    (and (not (at ?v ?f)) (at ?v ?t)
  (increase (electricity-used) (time-to-drive ?f ?t))) )
```

for the aggregated metric function:

```
(:metric minimize
    (+
      (* 10 (electricity-used))
      (+
        (* 1 (fuel-used))
        (* 1 (walked))
      )
    )
)
```

The cost of the *DRIVE-ELECTRICTRUCK* action is its impact on the plan metric. The plan metric θ=10$*$(electricity-used)+(fuel-used)+(walked). Using an operator Δ, which denotes the difference between the evaluation of the function before and after the action is applied, we can write the following formula for the impact of the action on the plan metric: $\Delta\theta = 10 * \Delta$(electricity-used)+$\Delta$(fuel-used)+$\Delta$(walked). This is equal to the difference of the plan metric value between the state where the action is applied and the state to which the action leads. We can compute the values for: Δ(electricity-used)=(time-to-drive ?f ?t), $\Delta(\text{fuel} - \text{used}) = 0$, $\Delta(\text{walked}) = 0$. This leads to the cost of the action being expressed as cost(DRIVE-ELECTRICTRUCK)$=$ 10$*$(time-to-drive ?f ?t).

When the metric fluent, (time-to-drive ?f ?t), is grounded, all its possible values are known in the initial state, and do not change during the plan execution. Therefore this action is compiled

into multiple durative actions, one for each of the values of (time-to-drive ?f ?t). One of the translated actions is:

```
(:durative-action drive-electrictruck_l0-l2
:parameters ( ?v - electrictruck ?d - driver)
:duration (= ?duration 100.0)
:condition
  (and (at start(at ?v s0)) (at start(driving ?d ?v))
       (at start(link s0 s2)) )
:effect
  (and (at start(not (at ?v s0))) (at end (at ?v s2))
       (at end (increase (electricity-used) 10.0)))
)
```

Similarly if the action cannot be fully grounded we obtain a context-dependent duration. For example, when translating the action:

```
(:action DRIVE-ELECTRICTRUCK
:parameters (?v - electrictruck ?f ?t - location ?d - driver)
:precondition (and (at ?v ?f)  (driving ?d ?v)(link ?f ?t))
:effect    (and (not (at ?v ?f)) (at ?v ?t)
  (increase (electricity-used) (* (load ?t) (time-to-drive ?f ?t)))) )
```

The impact on the plan metric can be expressed as: $10 * (\text{load}\,?t) *$(time-to-drive ?f ?t). Because (load ?t) can be altered by the load-truck action, and therefore is context-dependent. This part of the cost must be left until a state where it is applied is known. Therefore, in the compiled domain description, one of the actions obtained by compilation is presented below:

```
(:durative-action drive-electrictruck_l0-l2
:parameters ( ?v - electrictruck ?d - driver)
:duration (= ?duration (* 100.0 (load ?t)))
:condition
  (and (at start(at ?v s0)) (at start(driving ?d ?v))
       (at start(link s0 s2))
  )
:effect
  (and (at start(not (at ?v s0))) (at end (at ?v s2))
       (at end (increase (electricity-used) (* 10.0 (load ?t))))
  )
)
```

These examples illustrate that it is not always possible to fully ground the action prior to planning and therefore the planner is required to handle context-dependent action cost or, as it is in this case, action duration. All actions derived from the same action differ only in cost, expressed as duration.

4.2.2 Formal description of the algorithm

Formally speaking, the compilation takes as an input a metric problem description in the form of a tuple $< V, A, P, i, g, Cost(s_i, a_j) >$ where V is a set of metric fluents, A is a set of lifted actions (as described in PDDL domain), P is a set of propositions, i and $g \in \{P \cup V\}$ are initial and goal description respectively and can contain an assignment for some of the metric fluents from V. Cost(s_i, a_j) is the cost function returning a cost of a given action in a given state. For such a tuple the compilation returns a new temporal domain in PDDL, represented by $< V, T_a, P, i, g, Duration(s_i, a_j) >$, where T_a is a set of partially grounded temporal actions.

The compilation does not change the initial state or the goal. Duration(s_i, a_j) is the function returning a duration of a given action in a given state.

The compilation detects a set of invariants across metric fluents, denoted as V_i, and uses this information when generating compiled actions. For all actions $a \in A$, states s $\in P \cup V$, and a cost function cost(a, s), where the cost function does not depend on any metric fluent from V_i, it generates a temporal action a for which the duration(a, s) = cost(a, s). For all actions, a, where the cost function depends on some of the metric fluents from V_i, it enumerates all of the possible valuation of the set of metric fluents, on which the action depends, V_i^j, obtaining K valuations ($j = [1..K]$), and create K actions a^j where their durations are given by cost(a^j, s given V_i^j). For many actions the cost can be fully grounded, therefore: cost(a^j, s given V_i^j) = constant.

Using the information about assignment for metric fluents from the initial state the compilation infers the contribution to the increment of the plan metric from each of the actions, and if, by partially grounding an action, it is possible to determine the increase of the plan metric, the compilation partially grounds the action. It is possible for the compilation to fully ground some or all of the actions.

Actions for which the initial state contains insufficient information to determine cost, or the costs are context-dependent and cannot be calculated until the context in which the actions are applied is known, the algorithm leaves lifted. Actions, for which the initial state contains a partial information on the metric fluents required to ground an action, the algorithm partially grounds.

PDDL action is a tuple <*name, Param, Duration, Condition, Effect*> where *name* is the name of the action, *Param* is a set of parameters, *Duration* is a duration of the action, *Condition* is a set of preconditions that can contain propositions that must be true or bounds on metric values, *Effects* is a set of effects of the action which can be propositions that become true or false after the action is applied or increment or decrement effects on metric fluents.

Based on the increment or decrement of metric fluents which affect the plan metric, the algorithm infers the metric cost of the action as described previously in Section 4.2.1.1. This cost is then used as a duration for the temporal action. A temporal action is represented by a tuple *<name, Param, Duration, Start_Condition, End_Condition, Start_Effect, End_Effect>*, where Param - is the set of parameters, Start_Condition and End_Condition are preconditions for start and end of the action to be applicable, Start_Effect and End_Effect are the effects at start and end respectively. For the purpose of this work we do not require prevail conditions which must be satisfied during the execution of an action.

The translation can then be characterised as $\Gamma :< V,A,P,s,g > \rightarrow < V_t,T_a,P_t,s_t,g_t >$ such that $V = V_t, P = P_t, s = s_t, g = g_t$ and for every $action \in A$ the compilation creates a set of actions $\{action_t^1,...,action_t^{N_{a_t}}\} \in T_a$ such that, for a set of possible substitutions, $Sub \in Sub^*$ (this set might only include substitution for a subset of the parameters) $action_t^i$ is an effect of substitution Sub_i for *action*, such that:

- $action_t^i - duration \blacktriangleright$ is equal to the increment of plan metrics calculated based on effects of grounded (or partially grounded) *action*. Duration can be expressed as a function of metric fluents for a context-dependent cost.
- $action_t^i - Start\_Condition \blacktriangleright$ are all preconditions of *action*.
- $action_t^i - End\_Condition \blacktriangleright$ are empty.
- $action_t^i - Start\_Effects \blacktriangleright$ are all negative effects of *action*, as described in Section 4.1.2.5.
- $action_t^i - End\_Effects \blacktriangleright$ are all positive effects of *action*, as described in Section 4.1.2.5.

From the application point of view, the process only changes the PDDL domain and leaves the PDDL problem file unchanged. The planner has now information about both the plan metrics and plan metrics expressed as duration.

4.2.3 Summary of compilation from cost to temporal domains

This section introduces a novel compilation from cost to temporal domains. This compilation is a method for implementing the cRPG presented in Section 4.1. By exploiting some similarities between the construction of the TRPG and the cRPG, the compilation enables us to use an existing planners which build the TRPG as a proxy for the implementation of the cRPG.

A description of how each action is translated is presented, including a method for partially grounding actions to remove some complexity from the cost (duration) calculation. This simple technique allows the planner to quickly select an appropriate action and gives it more information about the action costs (durations).

For illustration purpose this section also contains examples of actions and their corresponding actions in the compiled, partially grounded, domain. Further evaluation of the impact of this compilation method on planners, and the quality of solutions, is presented in the next section.

4.3 Experiments and results

This section describes experiments carried out to evaluate the metric sensitivity of POPF2 and LPG in various configurations based on the ideas discussed and presents their results.

It begins with the investigation of how the compilation from metric to temporal domains impacts the quality and checks whether it enhances metric sensitivity as expected. Followed by the exploration of the impact of stochasticity on the quality of the results. We conclude with a discussion of the aforementioned approaches and their impact on quality of solutions and metric sensitivity of planners.

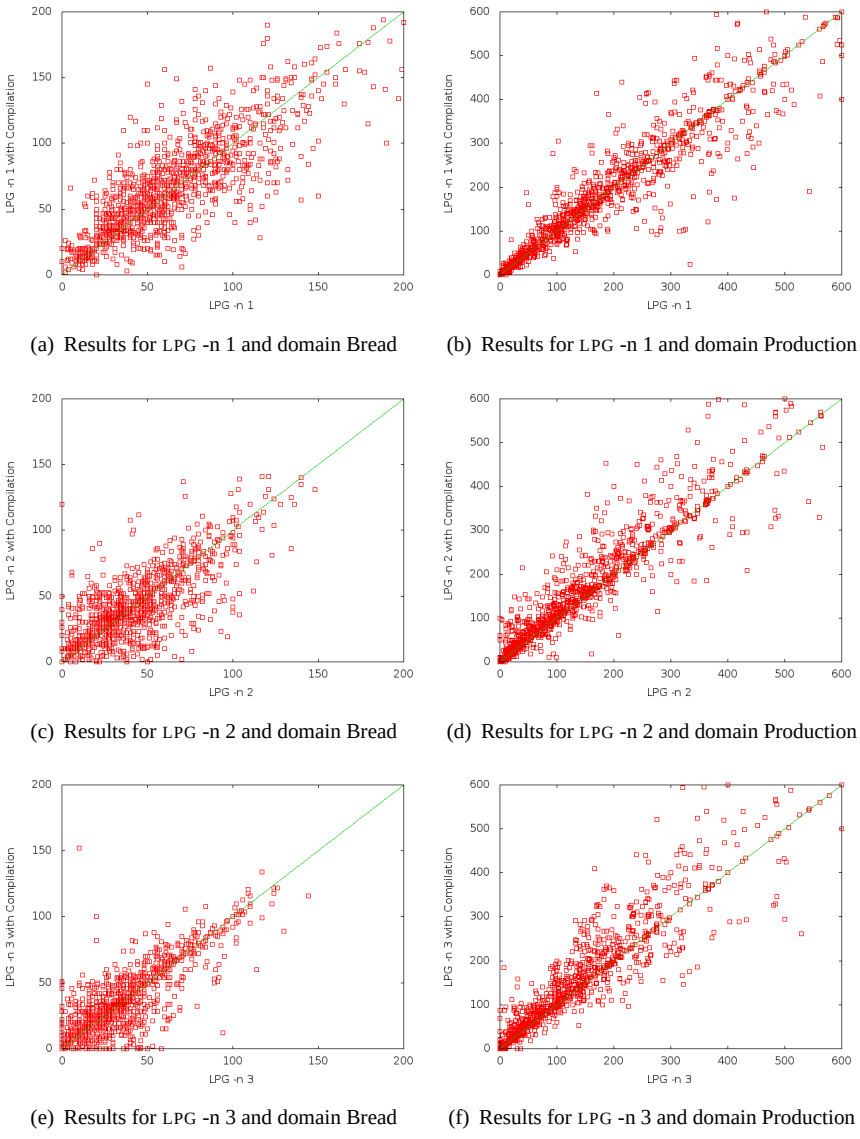
4.3.1 Compilation

In order to test how a compilation impacts on the quality of POPF2 and the metric sensitivity of planners we contrast it with results for LPG. The experiments are performed on the set of domains which exhibit interesting metric properties and, to some extent, decouple plan quality and plan length. These domains have been described in Section 2.5. They are also used in the previous experiment.

For the purpose of this experiment, similarly to the experiment on the impact of stochasticity, a set of problem files is generated from each problem file from each of the domains. Each of the problem files generated from the same problem file differs only in the plan metric function. To check the difference in generation of plans with and without the use of compilation, both POPF2 and LPG are run with compilation. In the results section we refer to the version of POPF2 using compilation as "POPF2 with compilation" or "metric sensitive POPF2", (MS-POPF2).

From each of domains a set of 66 plan metric functions is generated, equivalent to 66 solutions of a simplex A+B+C=10, for A,B,C all integers. The plan metrics for each weighting, depending on the domain are presented in Table 3.2.

Stochastic planners are run ten times for each weighting for each problem for each domain. The resulting quality of solution is then compared on a graph where the Y axis represents the values for compilation configuration and the X axis for the original results. Each image contains an X=Y line to separate the areas where compilation configuration generates better solutions, from the areas where compilation configuration generates worse solutions which coincides with below and above the X=Y line respectively. Therefore if more plans are shown below the X=Y line then more of the plans generated by compilation configuration for the same weights have better quality.



(a) Results for LPG -n 1 and domain Bread

(b) Results for LPG -n 1 and domain Production

(c) Results for LPG -n 2 and domain Bread

(d) Results for LPG -n 2 and domain Production

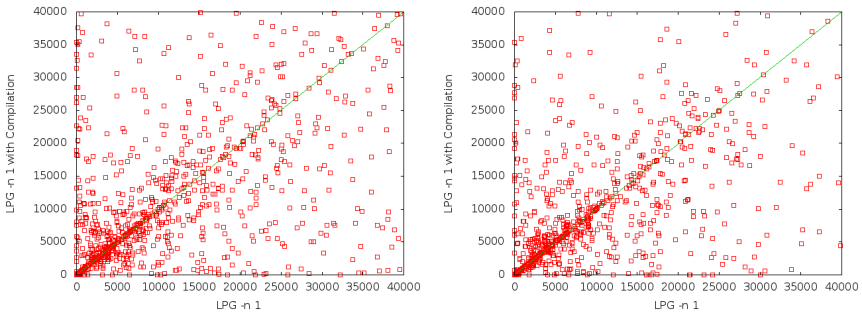
(e) Results for LPG -n 3 and domain Bread

(f) Results for LPG -n 3 and domain Production

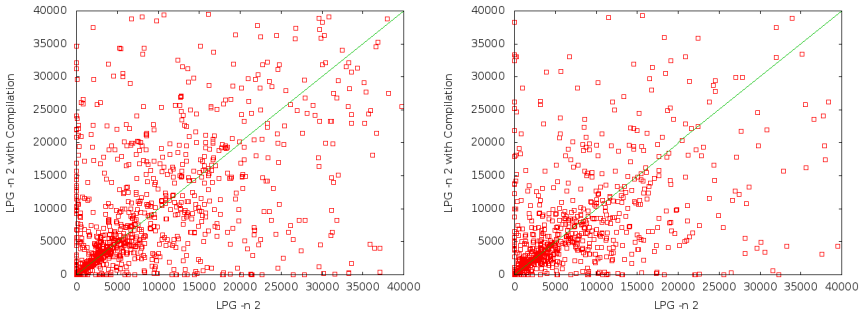
FIGURE 4.3: Impact of compilation results for all weights and all problems and weightings aggregated by domain. Images a), c), e) represent comparison of compilation versus original version of LPG -n 1, -n 2 and -n 3 respectively on domain Bread. Images b), d) and f) represent the same comparison for Production domain.

4.3.2 Results and discussion

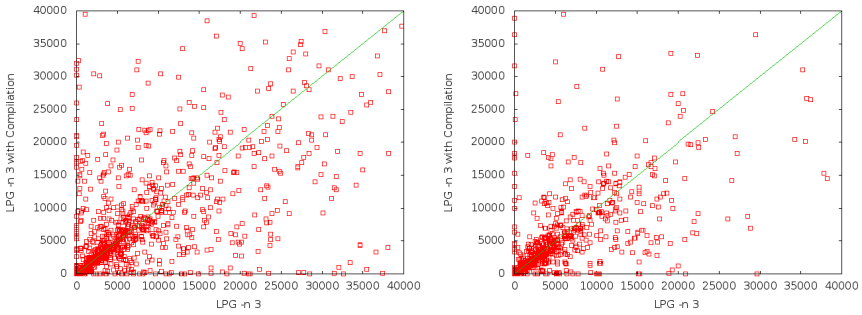
The results for the experiments described are presented in Figures 4.3, 4.4, 4.5 and 4.6.



(a) Results for LPG -n 1 and domain Driverlog (b) Results for LPG -n 1 and domain Driverlog metric



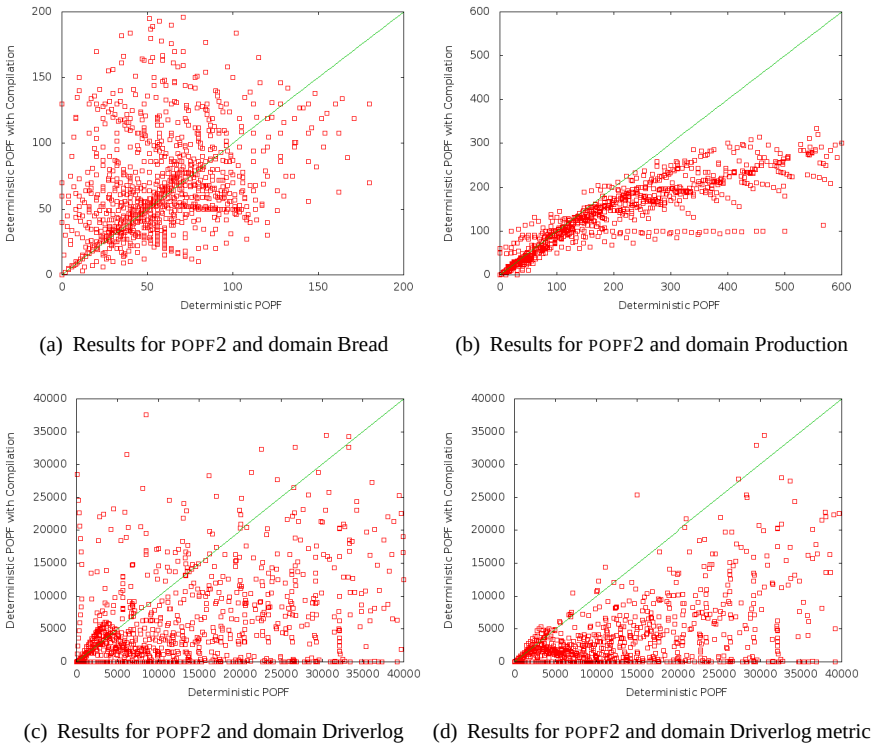
(c) Results for LPG -n 2 and domain Driverlog (d) Results for LPG -n 2 and domain Driverlog metric



(e) Results for LPG -n 3 and domain Driverlog (f) Results for LPG -n 3 and domain Driverlog metric

FIGURE 4.4: Impact of compilation results for all weights and all problems and weightings aggregated by domain. Images a), c), e) represent comparison of compilation versus original version of LPG -n 1, -n 2 and -n 3 respectively on domain Driverlog. Images b), d) and f) represent the same comparison for Driverlog metric domain.

The results suggest that there is no impact on the behaviour of LPG when presented with domains where the compilation was used to change the context-dependent cost to a duration of



(a) Results for POPF2 and domain Bread

(b) Results for POPF2 and domain Production

(c) Results for POPF2 and domain Driverlog

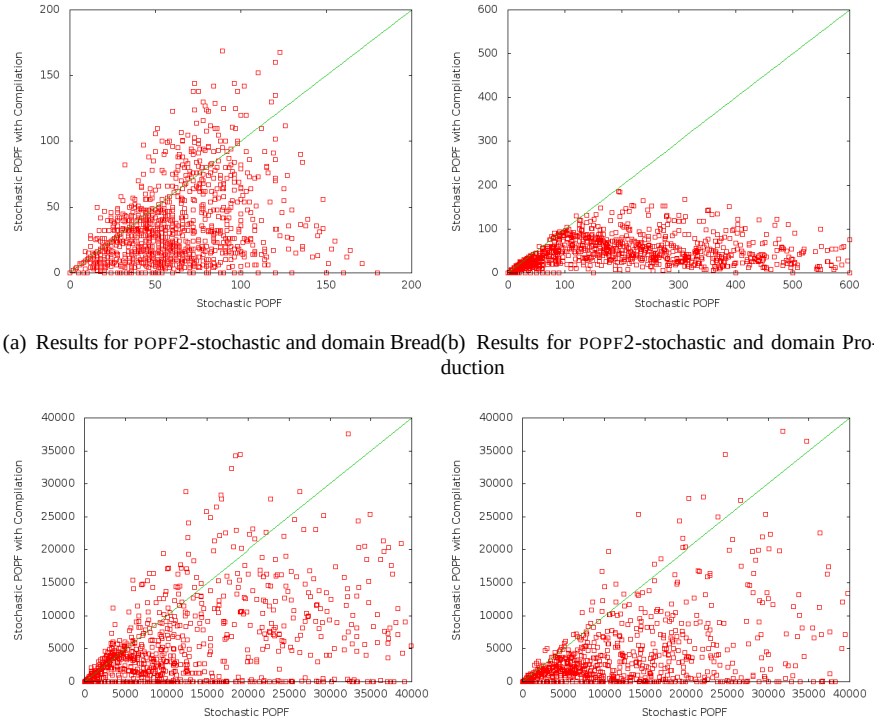
(d) Results for POPF2 and domain Driverlog metric

FIGURE 4.5: Impact of compilation results for all weights and all problems and weightings aggregated by domain. Images represent comparison of compilation and original version of POPF2 on domains Bread, Production, Driverlog and Driverlog metric pictured on images a), b), c) and d) respectively.

actions. This was to be expected as the compilation is designed to work best with POPF2 mechanism which, combined with the compilation, simulates the actual cost-RPG heuristic calculation.

In Figures 4.5 and 4.6 it is clear that the compilation significantly improves the quality of solutions.

What is more, when compared with the quality achieved by LPG, the quality of the solutions generated using the compilation with POPF2 performs better than LPG, which is a great improvement considering that we have started from a metric insensitive planner. This comparison is presented in Figure 4.7. An important fact to note is the large amount of solutions displayed



(a) Results for POPF2-stochastic and domain Bread

(b) Results for POPF2-stochastic and domain Production

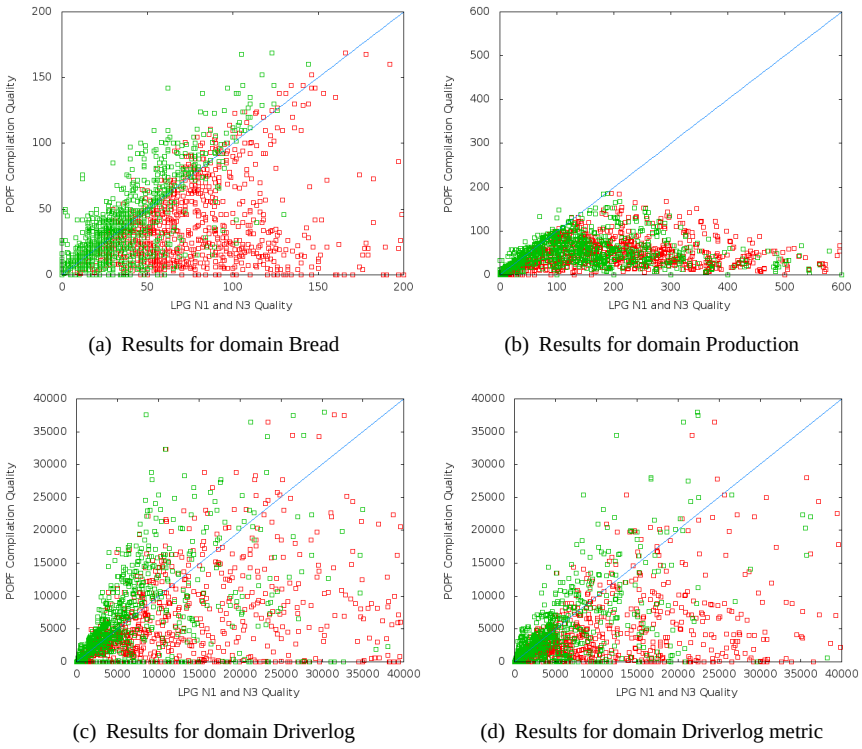
(c) Results for POPF2-stochastic and domain Driverlog

(d) Results for POPF2-stochastic and domain Driverlog metric

FIGURE 4.6: Impact of compilation results for all weights and all problems and weightings aggregated by domain. Images represent comparison of compilation and original version of POPF2 stochastic on domains Bread, Production, Driverlog and Driverlog metric pictured on images a), b), c) and d) respectively.

on the X axis. These are the solutions for which POPF2 found the best plan among all of the planners, and LPG found relatively bad solution. The large amount of these plans significantly affects the average quality score of both planners.

Aggregated numeric results for these experiments are presented in Appendix C. From the Appendix the most notable results are the comparisons of all of the approaches presented in Tables 7.3, 7.15, 7.11 and 7.7. These tables present averages of all of the results for each domain. The experiment is repeated ten times to show that the values are stable. The very big differences for Driverlog domains between the compilation and other approaches come from the fact that the compilation performs particularly well in finding the closest paths as demonstrated in the



(a) Results for domain Bread

(b) Results for domain Production

(c) Results for domain Driverlog

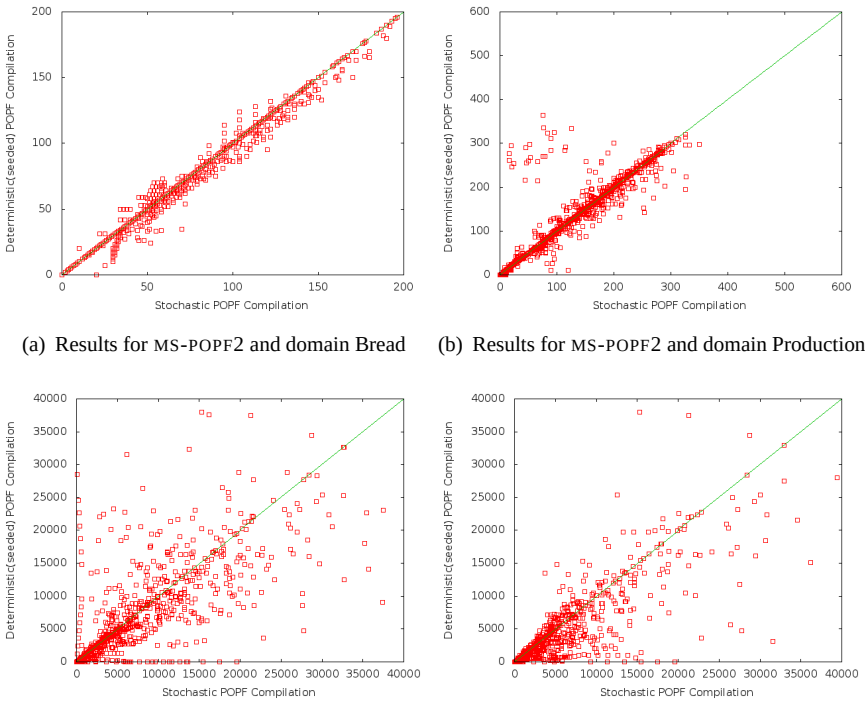
(d) Results for domain Driverlog metric

FIGURE 4.7: Impact of compilation results for all weights and all problems and weightings aggregated by domain. Images represent comparison of POPF2 with compilation and original version of LPG in configuration with -n 1, in red, and -n 3, in green, on domains Bread, Production, Driverlog and Driverlog metric pictured on images a), b), c) and d) respectively.

example above. This generates the best solution for many weightings, and LPG, due to its stochastic behaviour, in many cases finds much poorer solutions resulting in the relative score function returning huge values. When all of the weightings for which POPF2 returns best results are removed, the difference between LPG N3 and POPF2 is 2.65 and 1.69 respectively, which still evaluates in favour of POPF2 with compilation.

4.3.3 Impact of stochasticity on POPF2-compilation

This section describes an experiment which aims at measuring the impact of stochasticity on the metric sensitivity of a planner, where the metric sensitivity is obtained using the compilation described in Section 4.2. The purpose of this experiment is to show how the metric



(a) Results for MS-POPF2 and domain Bread

(b) Results for MS-POPF2 and domain Production

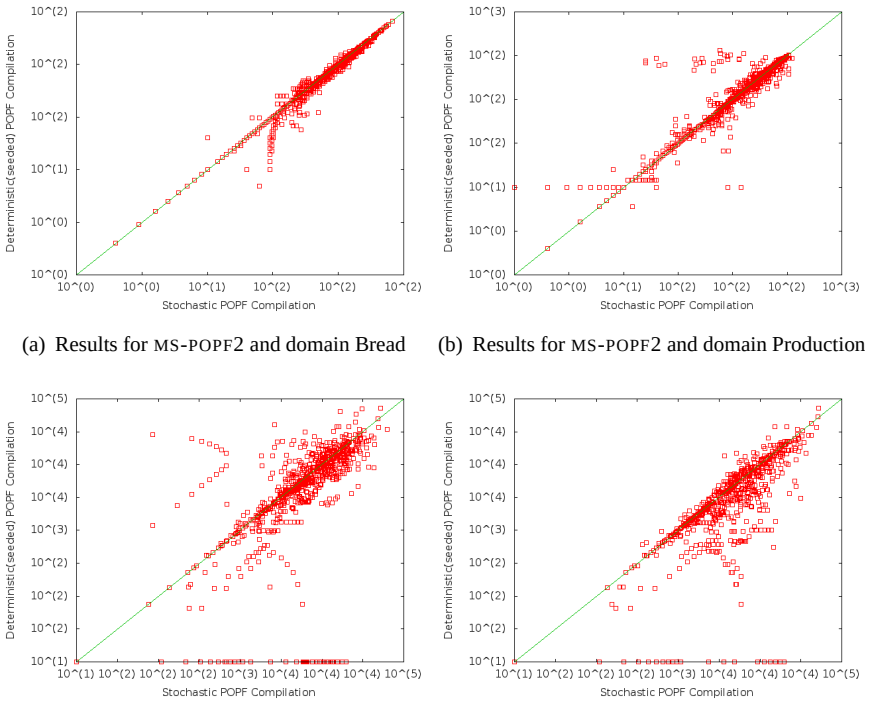
(c) Results for MS-POPF2 and domain Driverlog

(d) Results for MS-POPF2 and domain Driverlog metric

FIGURE 4.8: Results for all weights and all problems and weightings aggregated by domain. Images represent comparison of stochastic and deterministic version of POPF2 on domains Bread, Production, Driverlog and Driverlog metric pictured on images a), b), c) and d) respectively.

sensitivity of a planner is influenced by uninformed stochasticity. Two versions of POPF2 are used; original and with added stochasticity.

The changes to POPF2 are the same as described in Section 4.3.3 and include adding stochasticity in two places. First, in heuristic evaluation, and the second is branch ordering. The order in which actions are considered when expanding a state from which heuristic gives no clear guidance. It is important to note that opposed to LPG, POPF2 is a deterministic planner and the stochasticity which is added is not integrated with its search as well as in the case of LPG. Stochasticity in LPG is informed in a sense that it helps it in tie breaking and discovering areas of search space which would not normally be discovered, but might also be helpful.



(a) Results for MS-POPF2 and domain Bread

(b) Results for MS-POPF2 and domain Production

(c) Results for MS-POPF2 and domain Driverlog

(d) Results for MS-POPF2 and domain Driverlog metric

FIGURE 4.9: Logarithmic scale results for all weights and all problems and weightings aggregated by domain. Images represent comparison of stochastic and deterministic version of POPF2 with compilation on domains Bread, Production, Driverlog and Driverlog metric pictured on images a), b), c) and d) respectively.

The experiment is run for domains Bread, Production, Driverlog and Driverlog Metric. For each of the problems from each of domains, a set of 66 plan metric functions is generated, equivalent to the 66 solutions of a simplex A+B+C=10, for A,B,C all integers. The plan metrics for each weighting, depending on the domain, are presented in Table 3.2.

The results are present in Figure 4.8, on Page 87 using a linear scale and in Figure 4.9, on Page 88 , using a logarithmic scale to focus on different areas of the solution space.

Similarily to the experiments from Section 4.3.3, introducing stochasticity into POPF2 with the compilation did deteriorate the average quality of solutions. The plans did not suffer a big drop in quality. These results indicate that, when the search heuristic is cost oriented,

making stochastic decisions diverges the search from the path towards a high quality solution. This slight dip in the quality is visible in Figures 4.8 and 4.9 for linear and logarithmic scale respectively.

4.4 Conclusion

This chapter presents a new cost-RPG heuristic. It is based on an RPG approach, but by expanding the graph by cost rather than by layers the heuristic favours cheaper rather than shorter plans. Therefore, the new heuristics extends the class of problems which it can solve effectively from length correlated to monotonic cost (Definition 2.4). This approach is similar to the other cost-based RPG approaches presented by Sapena and Onaindia [54] and Fuentetaja et al [19]. The main difference between these approaches is how action cost in the cRPG is calculated, and whether the insertion of an action or the action effects is postponed by the cost. In this work we present a method which works with the context-dependent action costs.

The cost-RPG heuristic works particularly well, in comparison with the state-of-the-art, in finding plans where the number of actions does not correlate well with the change of cost. In the example presented in Section 3.1 there are two routes, A: $L0-L2-L5$ and B: $L0-L1-L3-L4-L5$. It is common for planners to favour the shorter action sequence, but more expensive, route A. In constructing the cost-RPG, the end of the action (drive-electrictruck ?v L0 L2 ?d) would not be added to the plan graph until after the cheaper route via $L1$ is already in place. As part of the heuristic we present a novel method of reasoning with context-dependent action cost. This is a very simple technique of calculating the action costs just before heuristic calculation starts, which proved to work well. The downside of this method is the fact that it fixes the action costs for the duration of heuristic function calculation. The benefits include an informed method of dealing with context-dependent action costs.

The comparison of performance shows that although cost-RPG requires more time to find a solution, it finds a solution of higher quality than LPG. The comparison is limited to LPG because most planners do not handle context/metric dependent action costs.

Results, presented in Section 4.3, show that the use of cRPG leads to a metric sensitive performance that generates plans of a higher quality than the closest competitor. We have shown that

LPG, as other planners, favours shorter plans over cheaper in terms of plan metrics. Our cRPG heuristic overcomes this limitation. The heuristic has been also implemented using a current temporal planner, POPF2, and a novel compilation from metric to temporal models.

The main contributions described in this chapter are:

- Cost-based RPG heuristic.
- Method for handling the context-dependent action cost by calculating cost in each state.
- Implementation of cost-RPG using a novel translation from cost to temporal models.

Chapter 5

Multi-objective planning and generation of APF of plans

Generation and evaluation of approximations of pareto frontiers (APF) is the main focus in this chapter. Current research in AI planning related to generation of Pareto Frontier (PF), APF and sets of plans is described. All of this work focuses on presenting the user with a selection of different and good quality plans. The meaning of a *different* and *good quality* plans is also discussed in the context of sets of plans. A survey of methods for evaluation of frontiers of plans is presented. Based on the methods surveyed, a selection is made of the most appropriate evaluation indicators for the purpose of comparison of multi-objective planning systems. A Multi-Objective Planning System (MOPS) is then described and evaluated with different configurations. The impact of stochasticity on the process of generating APF and the APF's quality is discussed. We present a method of exploiting stochasticity to further enhance the quality of APF. The chapter concludes with a discussion of these results in the context of the current state-of-the-art of multi-objective planning.

5.1 Current work

This section presents some current approaches to generating sets of solution diverse plans in response to a planning problem. The work presented here tackles the problem, or part of the problem, of answering the user's needs by generation and presentation of sets of diverse or high quality plans. It starts from MO-GRT, which introduces the notion of planning with multiple criteria. It follows up with the description of a recent research in generating high quality pareto frontier approximation sets using evolutionary algorithms. This is followed by a description of a work on generating sets, with set cardinality, of diverse plans where each of the plans must be distant from other plans by a given amount, as measured by a distance measure we define. This section concludes with a survey of related methods for the generation of pareto frontiers as used in operational research.

5.1.1 MO-GRT

GRT [51] is a domain independent heuristic STRIPS planner. It works in two phases, where the first phase estimates costs of arriving to the goal state from each of the facts. The cost is expressed in the number of actions that needs to be applied. During the search, the heuristic evaluation of a state is then based on the estimates of each of the facts from a given state and related facts, as defined by the authors. The multi-objective version of GRT, MO-GRT [52] is an extension of GRT to reason with multiple-objectives. It creates a tree of criteria (objectives), such that the leaves of the tree are basic criteria for which a measure is defined. All other nodes in the tree are a higher level criteria, called compound criteria, which are composed of an aggregation of basic and non-basic criteria. MO-GRT uses a weighted sum of criteria as the aggregation method. MO-GRT uses a multi-criteria A* search algorithm where the cost of the already found plan, and the estimate of the cost to the goal, are computed using two different criteria hierarchies. The cost for both of them is calculated bottom up, starting from the basic criteria and aggregating them into compound criteria.

Similarly to GRT, MO-GRT pre-calculates the cost of arriving to the goal from each of the facts. However, in multi-objective domains the cost is no longer a single number but a vector of costs, where the first cost is always plan length and is treated separately. In both single-objective and multi-objective GRT versions, the heuristic evaluation is inadmissible and is therefore not used for pruning.

The planner is evaluated using a modified Logistic domain and the criteria considered are length, cost and duration.

In their work, Refanidis and Vlahavas [52] introduce a very interesting way of hierarchically combining objectives. In theory, the aggregation function used does not have to be a weighted sum and therefore creates a possibility of obtaining a complex cost function representation. In practice, only a weighted combination of cost and time is used. This can be explained by the lack of multi-objective domains and the lack of expressive power (like metric fluents) of the language at that time.

5.1.2 Evolutionary algorithms

In their work, Khouadjia et al. [35] describe a multi-objective Divide-and-Evolve algorithm for the generation of pareto frontiers of plans. Evolutionary Algorithms (EAs) mimic processes which occur in natural evolution. In principle, EA comprises two stages: reproduction and survival. In the first stage the population of individuals is randomly expanded. This step explores new possibilities. The second step evaluates each of the new members of the population and allows only the fittest to survive. In planning, these two stages are equivalent to exploring the search space by random expansions of states and pruning states which are poor in terms of heuristic value.

During the evaluation phase of a multi-objective EA, not one but multiple objectives are used to evaluate the new members. The result of this evaluation should be similar to a single objective, a fitness function introducing a total ordering. Due to the fact that simple dominance, as defined

in Section 5.3, does not introduce total order, indicators are used to assign scores to the new population members.

An evolutionary Divide-and-Evolve (DAE) [3] planner is adapted to reason with multiple objectives. DAE$_X$ is used in configuration with a planner X. It uses the additional planner to solve sub-problems between intermediate states on the path from an initial state to a goal state. When a goal is reached, the sub-plans are compressed and the fitness function is calculated.

The multi-objective configuration of DAE, the MO-DAE$_{YAHSP}$, uses YAHSP as an internal planner. The modifications of the original algorithm are done in two main steps. First, the selection function is now multi-objective. In their work Khoudjia et al. use HVI, described in Section 5.3.1, for the evaluation due to its pareto-dominance completeness. The second is that the internal planner must now return values for all objectives.

MO-DAE$_{YAHSP}$ is evaluated on selected instances of modified ZenoTravel (introduced at the third International Planning Competition [38]), Elevators, and Openstacks domains. The evaluation of the very small number of problems shows that in most cases the MO-DAE approach is better than LPG with -n 1 option. For an instance of a problem from the Floortile domain LPG performs better, and for Elevators the resulting frontiers are equal. MO-DAE performs better on a problem instance from Openstacks and the MultiZeno domains, where MultiZeno domain is a multi-objective version of ZenoTravel domain.

Despite the claims that this approach can handle multi-objective domains, it only minimises total-cost and makespan of the plan. There is no support for a cost expressed in metric fluents. This is mainly due to the fact that DAE relies on the internal planner to calculate the values of each objective and YAHSP only deals with minimising the cost expressed in the total-cost and the makespan. The good quality of plans obtained is an artefact of the types of domains used for comparison.

5.1.3 dDISTkSETs

In this section we presents the work on dDISTANTkSETs [45] which offers a different approach at generating sets of plans. The sets and the method of generating them is described below. At the end of the section this approach is evaluated against a simple weight sampling approach, and the results are discussed.

Nguyen et al. [45, 46] propose an approach to the generation of sets of plans spanning across the search space. The idea is based on the assumption that the user cannot explicitly specify preferences. The output of the planning process is a dDISTANTkSET of plans, where a dDISTANTkSET contains k plans and each plan is at least a distance d from the other plans in the set. The distance is defined based on actions, causal links or the states visited, as described in Section 2.4.

LPG is used to generate the set of plans. One observation that follows this use is that LPG produces plans that are different according to the pressure placed on its search through the heuristic it uses to guide its local search behaviour. This pressure comes from the new way of evaluation of the output set of plans. The alteration is the use of Integrated Convex Preference (ICP), Definition 2.8. This measure is used inside LPG's heuristic instead of the standard execution cost. This set of plans is not intended to be optimal as the focus is on generating a variety of different solutions in terms of their plan distance measure. The good quality of solutions should come from the fact that LPG naturally searches for good quality plans.

The output set is evaluated based on ICP which is defined in Definition 2.8. For each plan from the set ϕ, a weighted sum of time t_{π_i} and cost c_{π_i} of that plan is calculated. For more details please refer to [45] section 3.2.

The ICP score is integrated into LPG and is used to drive its search towards plans different from the ones already found. This produces a good range of different plans. However, our experiments suggest that this is achieved at the cost of quality. Nguyen et al. are concerned with

generating distinct plans and not specifically the quality of those plans (although this is considered). Therefore the resulting sets presenting different solutions are usually of poor quality. The reason why the quality of the solution found is deteriorated by increasing its qualitative difference is that it is much easier for the planner to find qualitatively different solutions further from the pareto-optimal set than it is on it, or close to it. In a close proximity to the pareto-optimal set solutions are often close to each other (in terms of the metrics used by Nguyen et al.). When the solutions lie close together the pressure to find new and significantly distinct solutions forces the planner to ignore the high quality solutions from the close proximity of the ones already found. As a consequence the planner searches in different areas of the search space, further away from where the good quality solutions lie.

To demonstrate how the approach taken by dDISTANTkSETs compares to a simple weighted approach we have conducted the following experiment: for a problem from Driverlog domain, we run LPG with dDISTANTkSET's algorithm, and a sampling approach based on an aggregated metric function with different weights on two metrics. The metrics used in the experiment are $(electricity\text{-}used)$ and $(fuel\text{-}used)$. Again the modified version of Driverlog domain, described in Section 2.5.1, has been used.

The distance measure used by Nguyen et al is biased towards plans which appear different by using different actions, but which use exactly the same amount of resources. An alternative measure of distance between plans, based on the plan metric space, is shown in Definition 2.7 and is equivalent to the euclidean distance in the space described by the metrics. This definition considers plans to be distant if they have different values under one or more of the plan metrics. When evaluating the dDISTANTkSETs using this distance metric (an example output is shown in Figure 5.1) we observe that plans typically lie in the same place in the metric space. They are also, as expected, being pushed away from the area where good quality solutions lie due to the difficulty of finding good quality solutions distinct from each other. This reveals that the distance based on actions, causal links or states visited does not correspond to the distance

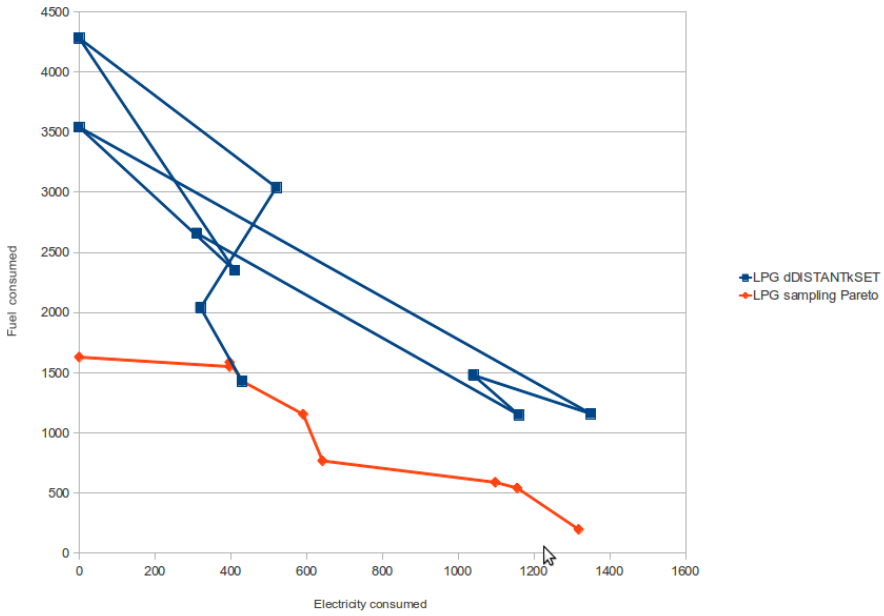


FIGURE 5.1: Difference between Pareto Sampling method and dDISTANTkSET generated using LPG. Connections indicate the order in which points were generated.

within an APF and, therefore, dDISTANTkSETs are not a good mechanism for generating APFs.

5.1.4 Mathematical methods for pareto frontier generation

Mathematical methods for calculating the pareto frontier are described in many papers such as [59], [31], [39] and [64]. They all explore various mathematical methods for calculating the pareto frontier. These methods are divided into 3 main categories depending on when the user preferences are known. These categories are: a priori articulation of preferences, a posteriori articulation of preferences and no articulation of preferences. Surveyed methods include various weighting approaches, lexicographic method, bounded objective function, goal programming, physical programming (PP), normal boundary intersection (NBI), normal constraint method (NC), and genetic algorithms. The most interesting methods are Normal Boundary Intersection and Physical Programming. The reason why these are the most interesting is that

they can be used to calculate an even distribution of points as in [44]. The algorithm for PP is further explained in [42]. The reason why most weighted methods do not work well, considering the even distribution of points on the pareto frontier is examined in [11]. In [11] Das and Dennis examine various cases of pareto frontier shapes (concave and convex) and point out cases where most weighted, linear, methods cannot find points on convex parts of pareto frontiers. Physical Programming [41, 42] is the method which allows to calculate an even distribution of points across the pareto frontier. One of its benefits is that the decision maker does not have to specify weights between functions. The decision maker expresses his preferences by giving bounds on resources within which he would like the resource consumption or the price to be.

For example, a decision maker can say that using 100 units of fuel is ideal, between 100 and 120 is desirable, between 120 and 160 is acceptable but undesirable and above 160 is unacceptable. PP uses this information, and therefore a DM is required to provide these bounds on resources. There are 8 classes of criteria classification in PP, divided into 2 main subclasses: soft and hard constraints. These classes are for soft constraints: *smaller is better*, *larger is better*, *value is better*, *range is better*, which favours smaller, larger, exactly X, and any X within the range values respectively. Similarly for the hard constraints we have *must be smaller*, *must be larger*, *must be equal*, *must be in range*. For each of the soft classes the DM is required to specify 6 ranges of preferences: *Ideal*, *Desirable*, *Tolerable*, *Undesirable*, *Highly Undesirable*, *Unacceptable*. For the hard criteria only 2 ranges are defined: *Acceptable* and *Unacceptable*. Then based on these preferences PP uses Linear Physical Programming Weight (LPPW) algorithm to compute weights. These weights are then used in a new LP problem which tries to minimise the deviation from the most desirable ranges. The actual algorithm of calculating these weights and then formulating the LP problem is different for each of the classes of criteria. It is ommited here because the discussion of all cases is out of the scope and focus of this thesis. It is important to note that in [44] Messac and Mattson describe a

slightly different way of using PP; they generate the weights rather than computing them and then, based on their weights, they are able to generate even spread of Pareto points.

5.2 Presentation of PF

Once an APF is constructed, the presentation of it for human evaluation is also a challenge. Once all of the plans are generated and the trade-offs are known, the main concern is how to communicate the alternatives in a clear way, allowing the user to understand the trade-offs and make appropriate decisions. There has been good progress in representation of solutions, presented in the work of Giuliano et al. [25, 32]. The difficulty in presenting the distribution of solutions greatly increases as we introduce more and more dimensions. Dealing with visualisation of up to 3D spaces is not very challenging, but as the dimensionality increases, it is far harder to display the results. The approaches taken include projections of the PF onto lower dimensions and presenting them as plots or histograms of objective values, or as explicit values.

As an example, consider an APF for the 5th problem from the Bread domain, Figure 5.2. The figure represents a 3D projection, on (energy), (labour) and (pollution) dimensions, of a set of solutions which approximates the PF. Any higher dimensional projection is very hard to picture and that is why projections are used. Even the 3D frontier is sometimes difficult to visualize well. In Figure 5.3 three one dimensional histograms and three projections onto a 2D space are present.

It is in some cases easier to spot tradeoffs by looking at lower dimensionality diagrams.

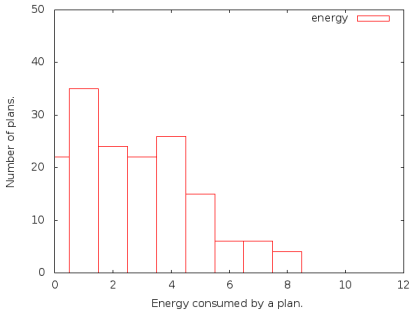
5.3 Evaluation and comparison of frontiers of plans

A qualitative evaluation of frontiers of solutions is a well explored topic in multiple fields, including Operational Research, Multi-objective Scheduling and Planning, and various fields

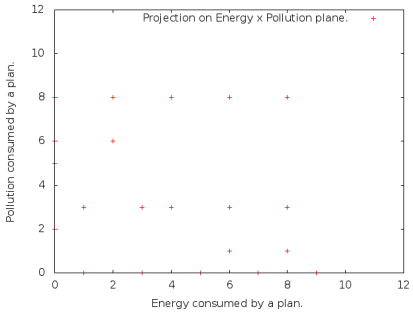
FIGURE 5.2: 3D APF for the fifth problem from the Bread domain. Projected onto a Pollution, Labout and Energy solution space.

in Engineering which deals with pareto frontiers. This section starts by introducing the notation used for classifying indicators. An indicator is a function which assigns a value to a frontier of solutions or a set of frontiers. The value assigned is regarded as the quality or relative quality, if a reference frontier is given. Examples of unary indicators for evaluation of the frontier of solutions and binary indicators for the comparison of two frontiers are then introduced. The section is concluded with a discussion of a survey of a wide range of indicators and their classification. Following Zietlers [65], we explain why a single indicator is not sufficient to determine which APF is better.

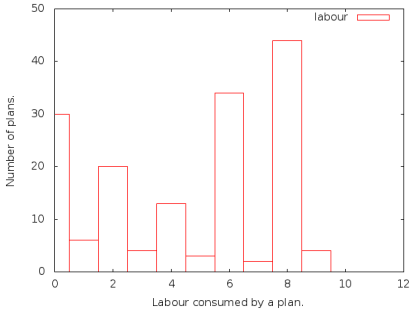
For the purpose of this section, let us denote an approximation of a pareto frontier as ϕ, which is a set of points, in our case plans π, $\pi \in \phi$. To evaluate a frontier means to assign a value to ϕ. This value can be used to directly compare it with other frontiers. Alternatively, a score can be assigned for one frontier in relation to an other. These measures are referred to as indicators.



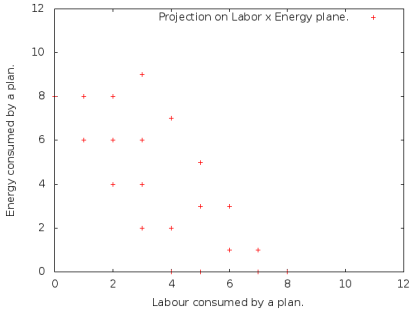
(a) Domain Bread, histogram of Energy metric.



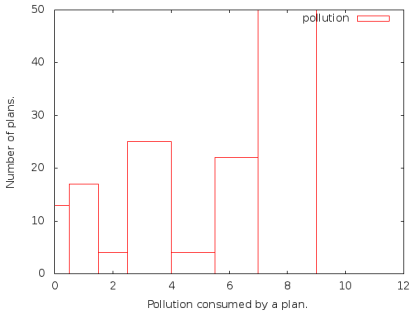
(b) Domain Bread, projection on Energy x Pollution plain.



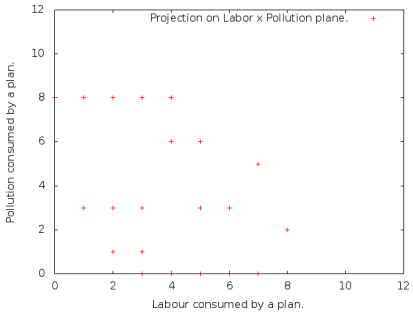
(c) Domain Bread, histogram of Labour metric



(d) Domain Bread, projection on Labour x Energy plain.



(e) Domain Bread, histogram of Pollution metric



(f) Domain Bread, projection on Labour x Pollution plain.

FIGURE 5.3: Projection of a 3D APF for domain Bread and problem 5. Images a, c and e present histograms of Energy, Labour and Pollution respectively. Images b, d and f present projections on two dimensional plans of Energy×Pollution, Labour×Energy and Labour×Pollution respectively.

Some indicators require two or more of the frontiers to be merged together. When merging

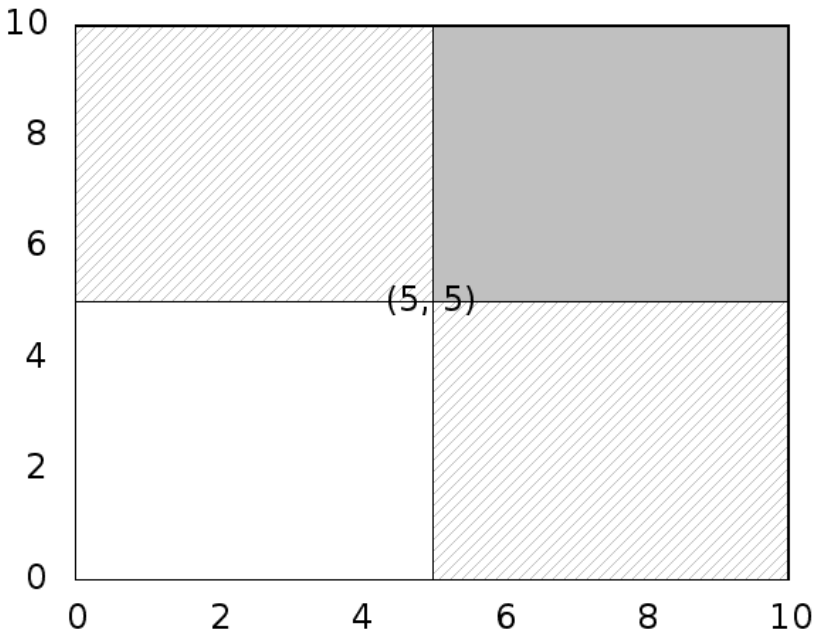


FIGURE 5.4: Diagram explaining domination relation. All the points within the blanc area dominate point (5, 5), all points within the dashed areas are equally good as (5, 5), and all points in the grey area are dominated by (5, 5). Points lying on lines (5, 5) to (5, 10) and (5, 5) to (10, 5) are weekly dominated by (5, 5). Points lying on lines (0, 5) to (5, 5) and (5, 0) to (5, 5) weekly dominate (5, 5).

frontiers it is important to remove dominated solutions. For this purpose, we define an operation of removing the dominated solution and an operation of adding (merging) pareto frontiers.

Definition 5.1. RemoveDominated(Π) is a function which, for a given set of plans, Π, returns a frontier of non-dominated solutions from Π.

$RemoveDominated(\Pi) = \{\pi : \pi \in \Pi \wedge \nexists_{\pi' \in \Pi} \pi' \succ \pi\}$

Where the plan domination relation $\succ$ is as defined in Definition 2.9 on Page 32.

Definition 5.2. Sum of two APF's: $\phi_1 \oplus \phi_2$, also referred to as merging two APFs, is:

$\phi_1 \oplus \phi_2 = RemoveDominated(\{\pi : \pi \in \phi_1 \vee \pi \in \phi_2\})$.

The comparison of single plans in multi-objective domains is as complex as comparing the entire APFs [65]. When comparing two plans in a multi-objective domain we need to consider

multiple metrics at the same time, and therefore it is not always easy to say that a particular plan is better than another. Figure 5.4 shows areas of the space where dominated, dominating, and equivalent solutions exist. Following Zietlers [65] we define:

Definition 5.3. (Relations between plans) Given some set of metrics θ for plans π_1, π_2:

- $\pi_1 \succ\succ \pi_2 \Leftrightarrow \pi_1$ strongly dominates $\pi_2 \Leftrightarrow \forall_{\theta_i \in \theta} \theta_i(\pi_1) < \theta_i(\pi_2)$

  i.e. when all objectives are better.

- $\pi_1 \succ \pi_2 \Leftrightarrow \pi_1$ dominates $\pi_2 \Leftrightarrow \forall_{\theta_i \in \theta} \theta_i(\pi_1) \leqslant \theta_i(\pi_2) \wedge \exists_{\theta_i \in \theta} \theta_i(\pi_1) < \theta_i(\pi_2)$

  i.e. when all objectives are no worse, and at least one is better,

- $\pi_1 \succeq \pi_2 \Leftrightarrow \pi_1$ weakly dominates $\pi_2 \Leftrightarrow \forall_{\theta_i \in \theta} \theta_i(\pi_1) \leqslant \theta_i(\pi_2)$.

  i.e. all objectives are no worse.

- $\pi_1 \parallel \pi_2 \Leftrightarrow \pi_1$ is not comparable to $\pi_2 \Leftrightarrow \exists_{\theta_i \in \theta} \theta_i(\pi_1) < \theta_i(\pi_2) \wedge \exists_{\theta_j \in \theta} \theta_j(\pi_1) > \theta_j(\pi_2)$.

  i.e. when some objectives are better and some are worse,

- $\pi_1 \rhd \pi_2 \Leftrightarrow \pi_1$ is better than $\pi_2 \Leftrightarrow \pi_1 \succeq \pi_2$ and $\pi_2 \not\succeq \pi_1$.

Definition 5.4. (Relations between frontiers) With respect to θ for frontiers ϕ_1, ϕ_2:

- $\phi_1 \succ\succ \phi_2 \Leftrightarrow \phi_1$ strongly dominates $\phi_2 \Leftrightarrow \forall_{\pi_2 \in \phi_2} \exists_{\pi_1 \in \phi_1} \pi_1 \succ\succ \pi_2$.

  i.e. when each plan from ϕ_2 is strongly dominated by at least one plan from ϕ_1,

- $\phi_1 \succ \phi_2 \Leftrightarrow \phi_1$ dominates $\phi_2 \Leftrightarrow \forall_{\pi_2 \in \phi_2} \exists_{\pi_1 \in \phi_1} \pi_1 \succ \pi_2$.

  i.e. when each plan from ϕ_2 is dominated by at least one plan from ϕ_1,

- $\phi_1 \succeq \phi_2 \Leftrightarrow \phi_1$ weakly dominates $\phi_2 \Leftrightarrow \forall_{\pi_2 \in \phi_2} \exists_{\pi_1 \in \phi_1} \pi_1 \succeq \pi_2$.

  i.e. when each plan from ϕ_2 is weakly dominated by at least one plan from ϕ_1,

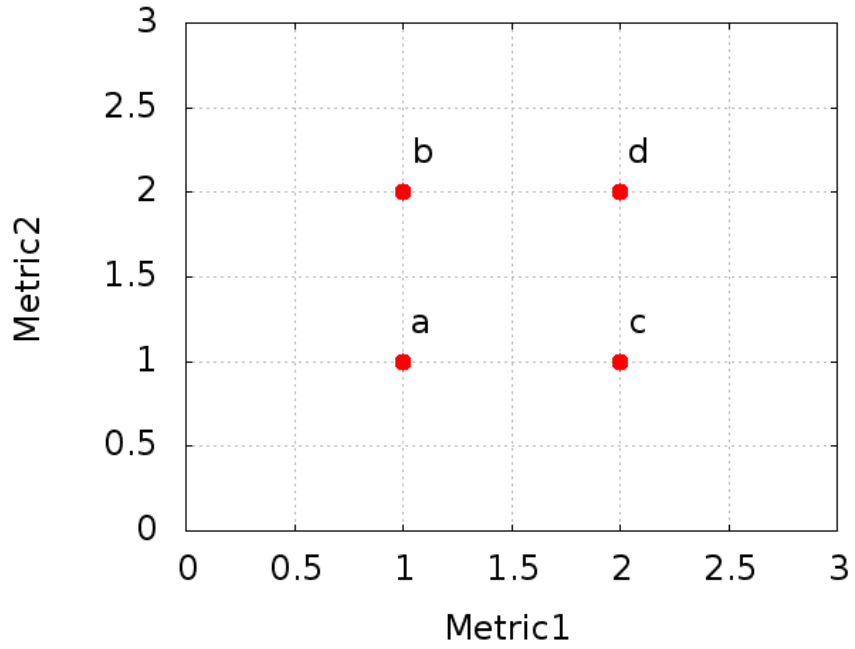


FIGURE 5.5: Diagram representing dominance relation for single plans in metric space. Based on the figure image we can say that: $a \succ\succ d$, $a \succ d$, $a \succeq d$, $a \succ b$, $a \succeq b$, $a \succ c$, $a \succeq c$, $a \succeq a$, $b \parallel c$, $b \succ d$, $b \succeq d$, $b \succeq b$, $c \succ d$, $c \succeq d$, $c \succeq c$, $d \succeq d$

- $\phi_1 \parallel \phi_2 \Leftrightarrow \phi_1$ not comparable or equally good to $\phi_2 \Leftrightarrow \exists_{\pi_1 \in \phi_1, \pi_2 \in \phi_2} \pi_1 \succ \pi_2 \wedge \exists_{\pi_1 \in \phi_1, \pi_2 \in \phi_2} \pi_2 \succ \pi_1$.

  i.e. when some plans are better and some are worse

- $\phi_1 \rhd \phi_2 \Leftrightarrow \phi_1$ better than $\phi_2 \Leftrightarrow \phi_1 \succeq \phi_2 \wedge \phi_2 \not\succeq \phi_1$.

  i.e. when ϕ_1 is no worse than ϕ_2, but ϕ_2 is not no worse than ϕ_1 and $\phi_1 \neq \phi_2$.

To better illustrate these relations, let us consider Figure 5.5 for plans, indicated by dots. The coordinates of plans a, b, c, and d are (1, 1), (1, 2), (2, 1), (2, 2) respectively. Based on the above definitions the following describes the relationship between plans: $a \succ\succ d$, $a \succ d$, $a \succeq d$, $a \succ b$, $a \succeq b$, $a \succ c$, $a \succeq c$, $a \succeq a$, $b \parallel c$, $b \succ d$, $b \succeq d$, $b \succeq b$, $c \succ d$, $c \succeq d$, $c \succeq c$, $d \succeq d$.

To illustrate how these relations behave for frontiers of plans please refer to Figure 5.6. Some examples of this relations include: PF $\succ\succ$ APF1, PF $\succ$ APF1, PF $\succeq$ APF1, PF $\rhd$ APF1, APF1 $\succ\succ$ APF3, APF1 $\succ$ APF3, APF1 $\succeq$ APF3, APF1 $\rhd$ APF3, APF2 $\succeq$ APF3, APF2 $\rhd$

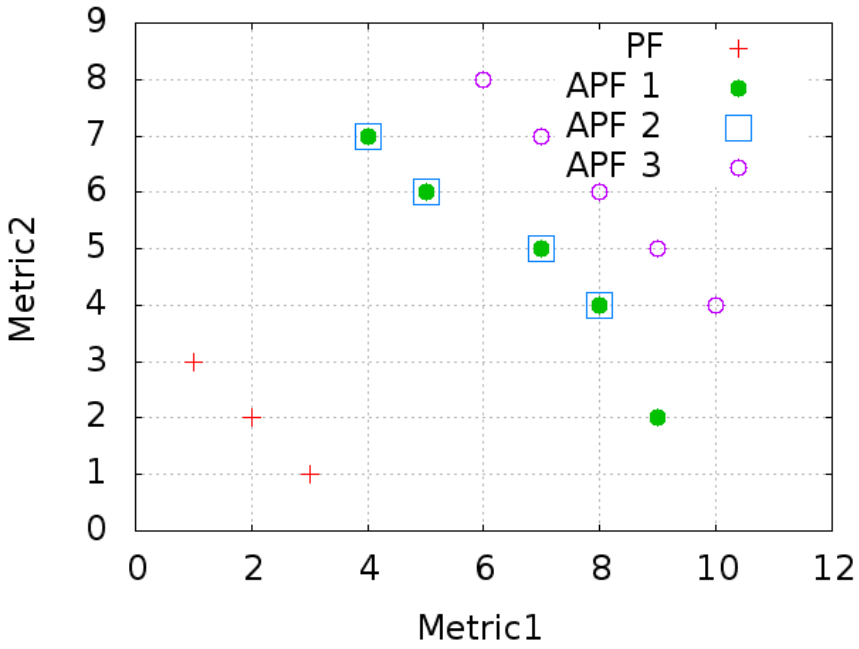


FIGURE 5.6: Diagram explaining domination relation for frontiers of plans. We can say the following about the frontiers from the image PF
$\succ\succ$ APF1, PF $\succ$ APF1, PF $\succeq$ APF1, PF $\rhd$ APF1,
APF1 $\succ\succ$ APF3, APF1 $\succ$ APF3, APF1 $\succeq$ APF3, APF1 $\rhd$ APF3,
APF2 $\succeq$ APF3, APF2 $\rhd$ APF3,
APF1 $\succeq$ APF2, APF1 $\rhd$ APF2,
And also APF1 $\succeq$ APF1, APF2 $\succeq$ APF2, APF3 $\succeq$ APF3.

APF3, APF1 $\succeq$ APF2, APF1 $\rhd$ APF2, APF1 $\succeq$ APF1, APF2 $\succeq$ APF2, APF3 $\succeq$ APF3. What is important to notice is that, although APF1 $\succ\succ$ APF3, the following is *not* true: APF2 $\succ\succ$ APF3. This is because the plan in APF3 with coordinates (10,4) is not strongly dominated by any plan in APF2.

Before we can present the indicators, two important definitions are necessary. The compatibility and completeness with the above relations.

Definition 5.5. Indicator I **is $\blacktriangleright$-compatible**, for an arbitrary binary relation $\blacktriangleright$, if:

$I(A) > I(B) \Rightarrow A \blacktriangleright B$ or $I(A) > I(B) \Rightarrow B \blacktriangleright A$

Definition 5.6. Indicator I **is $\blacktriangleright$-complete**, for an arbitrary binary relation $\blacktriangleright$, if:

$A \blacktriangleright B \Rightarrow I(A) > I(B)$ or $B \blacktriangleright A \Rightarrow I(A) > I(B)$

These two are used when discussing the properties and usefulness of indicators. A desired property of an indicator is that the comparison of its values for two frontiers gives information about the relation between the frontiers evaluated. Now we present a selection of methods used in literature to evaluate the quality of APFs.

5.3.1 Unary indicators

Unary indicators assign a single value to a single frontier. They evaluate a frontier based on the plans contained in the frontier, and do not use any additional information about other frontiers or plans. Unary indicators flatten all aspects of a frontier into a single number. It is very hard to find a descriptive and informative way of compressing an entire frontier into a single number. Many indicators have been developed to capture various aspects of the frontier. Below we present just a small selection of them. Further in this chapter we show how a comparison of frontiers can utilize multiple unary indicators to obtain better comparison results, in terms of $\rhd$ compatibility and completeness.

Average distance to a reference point This is one of the simplest methods for comparing the frontiers. For a given frontier ϕ and a reference point γ, we calculate a Cartesian distance between each point from the frontier to the reference point and take their average as the score.

Definition 5.7. Average distance metric $I^{avg}(\phi) = \frac{\sum_{\pi_i \in \phi} |\pi_i - \gamma|}{|\phi|}$ Where $|\pi_i - \gamma|$ is a distance measure defined in 2.7 and $|\phi|$ is the cardinality of the set of plans, ϕ.

When comparing frontiers of plans, the above measure is not affected by the distribution of the frontiers. This means that if two frontiers are in two different places of the space, they are going to be compared in the same way, as if they were in the same one. Figure 5.7 a) shows

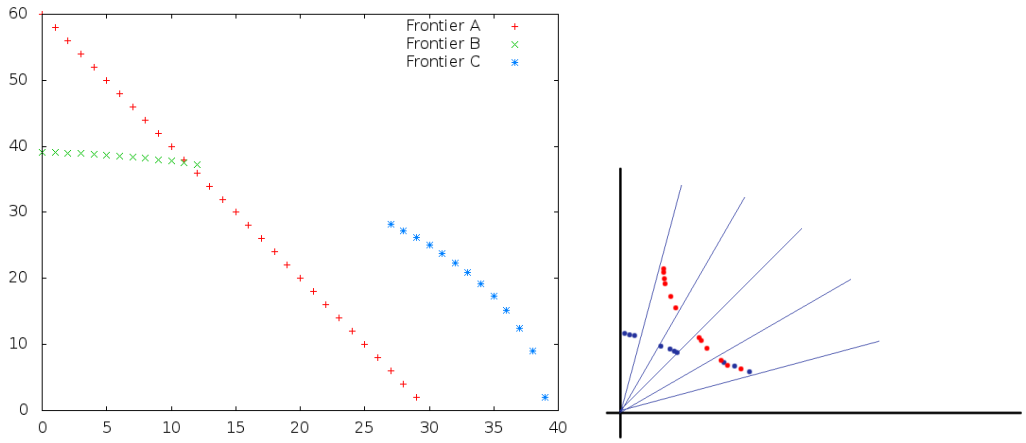


FIGURE 5.7: Examples of frontiers

three frontiers which would be assigned the same average score, although we can see that they are very different. In many cases plans from distinct areas of the metric solution space carry a different value to the user. The benefit of including them in the frontier is not comparable. A measure which ignores the placement of plans in the metric space, and their relations, is loosing this information. Therefore this metric carries the danger of assigning a value to an incomparable frontier such as frontier B and C from the Figure 5.7 a).

For the purpose of comparing the frontiers which lie in different areas of the solution space, and to include the notion of coverage and distribution into the quality metric, a new metric is proposed. The metric calculates the average distance from a reference point (typically the point 0), as shown in Figure 5.7 b), but this time the calculation is done separately for each of the areas of the search space. The areas are determined by cuts originating in the reference point, and their number vary depending on the method and application. For the comparison of two frontiers, numbers from the same sections are taken.

Hyper-volume indicator A very popular measure used to evaluate frontiers is the Hyper-volume Indicator (HVI) which is equivalent to Klee's measure, first formed by Victor Klee in his article [36]. This problem was then solved by John Bentley in 1977, whose notes are

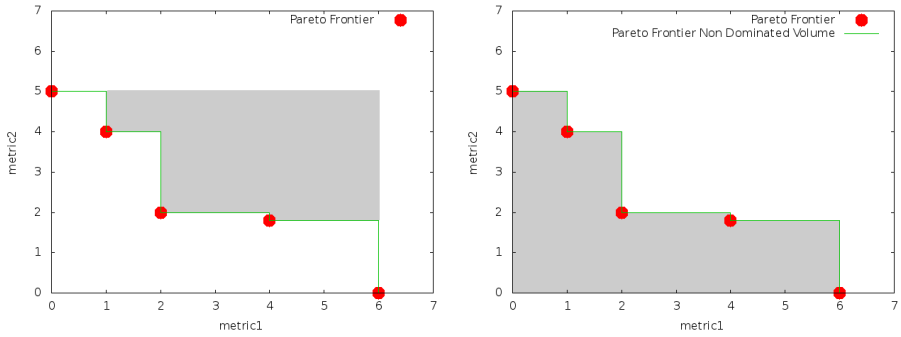


FIGURE 5.8: Graphical description of 2d Hyper-Volume Indicator and an example of 2d Non-Dominated Hyper-Volume Indicator.

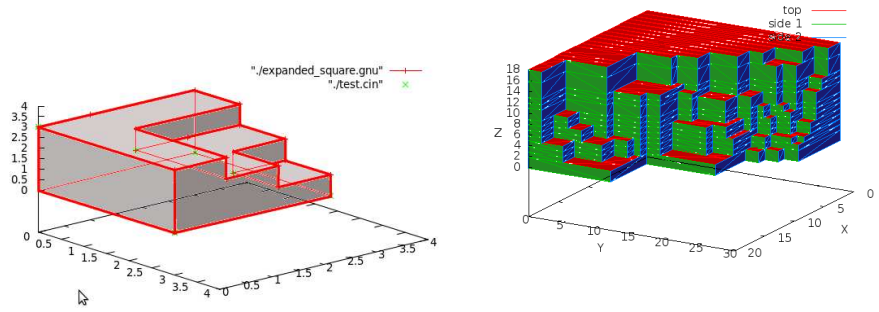


FIGURE 5.9: Graphical description of two 3d Non-Dominated Hyper-Volume Indicators.

unpublished. There have been many improvements to the algorithm [5, 7]. It is proven that for one and two dimensional spaces the optimal algorithm to calculate the HVI takes O($n$log(n)). For higher dimensions there exists an upper bound O($n^{d/2}$) [7]. However, no information on faster algorithms exists. Figure 5.8 shows an example of the area covered by the HVI.

Non-dominated hyper-volume indicator Similar to the HVI, we present a Non-dominated Hyper-volume Indicator (NDHVI) which conceptually is more appropriate for the planning context. The NDHVI is calculated based on Definition 5.8.

Definition 5.8. Non-Dominated Hyper-Volume Indicator (NDVHI) for nadir point γ, utopia point $\overrightarrow{u} = \overrightarrow{0}$ and APF = Π is $I^{NDHVI}(\gamma,\Pi) = V(\gamma) - I^{HVI}(\gamma,\Pi)$ Where $V(\gamma)$ is the volume enclosed by the nadir point.

NDHVI is represented in Figure 5.8 b) and Figure 5.9. It is an analogous measure to HVI, however, it is easier to understand and more intuitive in terms of planning APFs. When calculated it describes the hyper-volume of the part of the metric space which contains plans that are not dominated by any of the plans from already found APF. Consequently, this is the hyper-volume by which it is possible to increase the quality of the frontier.

What is more, it has the property that, when adding plans to a frontier (or merging two frontiers), the NDHVI can only decrease. Proof for this property is shown below.

Theorem 5.9. ***NDVHI is monotonic when adding plans to the frontier***.

Proof. Let us consider a frontier of plans Π and a plan $\pi^{'}$. There are three possible cases:

1. $\pi^{'}$ is dominated by at least one plan from Π

   When the plan $\pi^{'}$ is added to the frontier, it is, at the same time, automatically removed because it is dominated, and there cannot be dominated plans on the frontier. Therefore the $I^{NDHVI}(\gamma,\Pi) = I^{NDHVI}(\gamma,\Pi \cup \pi^{'})$

2. $\pi^{'}$ is dominating at least one plan from Π

   When $\pi^{'}$ is added, the dominated plans are removed from Π and the new score is calculated. The new plan $\pi^{'}$ must lie on the grey area of the NDHVI from Figure 5.9. Adding it to the frontier is equivalent to "cutting out" part of the frontier. Plans dominated by $\pi^{'}$ lie within the area 'cut out' from the NDHVI, therefore the NDHVI decreases.

3. $\pi^{'}$ is neither dominated nor dominating any plan from Π. This case includes $\pi^{'}$ weakly dominated/dominating some plans in Π.

   When $\pi^{'}$ is weakly dominated by one of the plans from Π, or weakly dominates one, $\pi^{'}$ lies on the border between NDHVI and HVI, and adding $\pi^{'}$ has no impact on the NDHVI value.

Based on the above, we can say that NDHVI is monotonic when adding new plans to the frontier. The same holds when merging two frontiers, as this is equivalent to sequentially adding all plans from the first frontier to the second.

□

Generation time Time of generation is an important metric when comparing frontier generators. It is the metric which directly impacts on how users interact with the system. Quick generators can provide ad hoc results while slow ones require user to wait for an answer. This might be crucial for the decision on which one to use.

In this work, the generation time is the time measured from the time the generator starts to work until the time when it produces a frontier of plans.

Distribution Even distribution of solutions along the frontier in the utility space is also very important. By "evenly distributed set of points" we mean a set where none of the areas is over or under populated.

The example measure of distribution is present by Ziadloo and Ghamsary [62]. The solution space is divided into equal size areas and the distribution is calculated based on the number of solutions which exist in each of the areas of the utility space.

An alternative metric is presented by Messac and Mattson [43]. For each point on the frontier we compute the radius of the smallest and largest spanning circle (Definition 5.10).

Definition 5.10. Spanning Circle, for a set of solutions Π and a point $\pi \in \Pi$, is a circle that does not enclose any $\pi_i \in \Pi$, π lies on the circle, and there exists a point $\pi' \in \Pi$ that lies on the opposite side of the circle. For a plan, π_i, examples of smallest, d_l^i, and largest, d_u^i, spanning circles for a point on a frontier are presented in Figure 5.10.

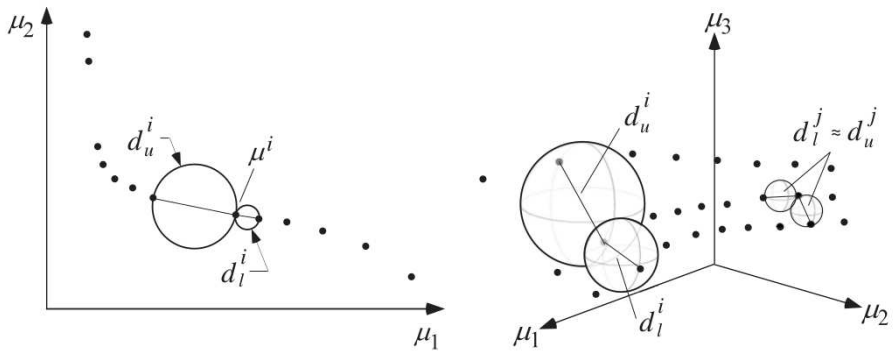


FIGURE 5.10: Graphical description of factors contributing the evenness of a distribution of points in 2D and 3D. Source: [43]

The variation coefficient of the lengths of the radius of those circles is our distribution measure, denoted as $I^{Distribution}$. Where the variation coefficient is the standard deviation divided by the mean.

Definition 5.11. APF Distribution is defined as $I^{Distribution}(\phi) = \frac{\sigma_d}{\hat{d}}$
Where σ^d is the standard deviation and $\hat{d}$ is the mean of the radii of the smallest and largest spanning circles for all points on the APF as presented in Figure 5.10.

Coverage and size The motivation behind this measure is to estimate the degree to which the sample of plans covers the shape of the pareto frontier. Due to the lack of knowledge about the shape of the actual pareto frontier being sought, it is not possible to give an exact measure of coverage. However, for the purpose of comparison of alternative APFs, we can say that, of two APFs with the same distribution measure, the one with larger distances between points or with more points will have better coverage. Therefore, as a proxy for coverage, we present two metrics. The first is the average size of the radius of circles used for calculating the distribution as in Definition 5.11. The second is the number of points on the frontier generated. Together these two measures give a good indication of the coverage of the pareto frontier offered by an APF.

In general, for the same value of the average radius, more plans mean a better coverage, but fewer plans often lead to a better distribution across the space. This reflects the difficulty in finding plans uniformly distributed across the entire pareto frontier. Therefore, as a proxy for coverage, the following indicators are used:

Definition 5.12. Average Radius of spanning circles, $I^{Avg(R)} = \frac{\sum_{i=[0..N]} d_l^i + d_u^i}{2*N}$.

Definition 5.13. Variance of a radius of spanning circles, $I^{Var(R)} = \frac{Var(d_l)+Var(d_u)}{2}$.

Definition 5.14. Size of the frontier, $I^{\#-plans} = |\Pi|$.

Definition 5.15. Time of generation of the frontier, I^{time}, is the total time taken to generate the APF measured from the point a planner is started, until it finishes execution.

5.3.2 Binary indicators

Below two binary indicators are presented. These indicators compare two frontiers together and the value which they compute describes the relation of the two frontiers. These indicators can convey more information about the relations between two or more APFs than comparing the values of unary indicators.

ε-indicator I^{ε} [65] is a qualitative measure which describes a distance between two frontiers. More precisely, it calculates a factor by which an APF is worse than another. The value of $I^{\varepsilon}(\phi_1, \phi_2)$ is equal to the smallest factor ε by which any point from ϕ_2 is dominated by a point in ϕ_1. The factor of domination between pair of plans $\pi_1 \in \phi_1$ and $\pi_2 \in \phi_2$, for a metric vector $\theta(\pi)$ = $[\theta_1(\pi), ...\theta_N(\pi)]$, is given by the smallest ε such that: $\forall_{i=[1..N]} \theta_i(\pi_1) \leqslant \varepsilon * \theta_i(\pi_2)$. Therefore the ε-Indicator is defined as:

Definition 5.16. ε-Indicator of two frontiers ϕ_1 and ϕ_2 is given by:
$I^{\varepsilon}(\phi_1, \phi_2) = \arg\min_{\varepsilon} \forall_{i=[1..|\phi_1|], j=[1..|\phi_2|], k=[1..N]} \theta_k(\pi_1^i) \leqslant \varepsilon * \theta_k(\pi_2^j)$ where π_1^i and π_2^j are i-th and j-th plans from the frontier ϕ_1 and ϕ_2 respectively.

Binary F-indicator F-Indicator is a binary indicator which for two APFs ϕ_1 and ϕ_2 returns the fraction of ϕ_1 which occurs in combined APF. It can be calculated as:

Definition 5.17. F-Indicator of two frontiers ϕ_1 and ϕ_2 is given by:
$I^F(\phi_1, \phi_2) = \frac{|\phi_1 \cap (RemoveDominated(\phi_1 \cup \phi_2))|}{|RemoveDominated(\phi_1 \cup \phi_2))|}$.

5.3.3 Discussion of indicators

In their work, Zietzler et al [65] present a classification of a comparison method for APFs derived from a set of indicators. To define a classification method we first need to define an evaluation method as follows:

Definition 5.18. Evaluation Method, also called an interpretation method, is a function which, for a set of indicators, returns a boolean. Formally denoted as: E($I_1, ...I_N$) = $true$ or $false$. Where I_i is a vector of values for indicator I_i for all evaluated frontiers.

The evaluation method for two frontiers ϕ_1 and ϕ_2, and a set of two indicators I_1 and I_2, could be used to determine whether the first frontier is better than the second. A definition of such comparison method is: E(I_1, I_2) = true if $I_1^{\phi_1} < I_1^{\phi_2}$ and $I_2^{\phi_1} < I_2^{\phi_2}$, false otherwise.

Using the evaluation method and a set of indicators a comparison method aims to answers the question whether the first frontier is or is not better than the second.

Definition 5.19. Comparison Method: For two APF's ϕ_1 and ϕ_2, a set of indicators I = $I^1, ..I^N$ and an evaluation function E(I), a comparison method is defined as:
$C_{I,E}(\phi_1, \phi_2) = E(I(\phi_1), I(\phi_2))$ Where $I(\phi_i)$ is a vector of values of each of the indicators from I $I^1(\phi_i), ...I^N(\phi_i)$.

Ideally, a comparison method could be constructed for each of the relation operators. This would allow to sort frontiers automatically. The most interesting relation is to determine whether a frontier is better than the other: $\phi_1 \rhd \phi_2$. The question which we try to answer

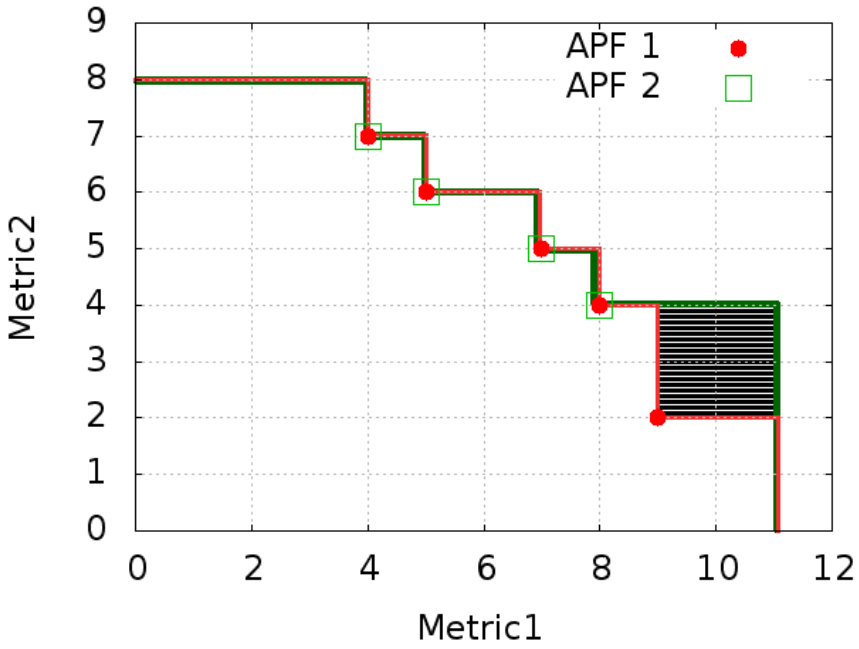


FIGURE 5.11: Difference between NDVHI for APF1 and APF2 from Figure 5.6. Nadir point (11, 8) was used for the construction of NDHVI.

is whether we can create such comparison method which would be both $\triangleright$-compatible and $\triangleright$-complete. Let us look at some examples of comparison methods.

An example of a comparison method is taking the values of NDHVI and concluding that a frontier with a lower value is better. We would denote that as $C_{I^{NDHVI},E^<}(\phi_1,\phi_2)$. We say that this comparison is $\triangleright$-complete because $APF1 \triangleright APF2 \Rightarrow I^{NDHVI}(APF1) < I^{NDHVI}(APF2)$ but it is not $\triangleright$-compatible. To show that let us consider APF1 and APF2 from Figure 5.6 and their NDHVI shown in Figure 5.11. The difference between NDHVI area is indicated by a dashed background. By calculating the NDHVI we obtain $I^{NDHVI}(APF1) < I^{NDHVI}(APF2)$, but $APF1 \triangleright APF2$. This indicator is also $\not\triangleright$-complete, meaning that if $I^{NDHVI}(APF1) < I^{NDHVI}(APF2)$ then we know that $APF2 \not\triangleright APF1$.

Many indicators which do not directly assess the quality of the frontier such as time, distribution or the number of plans on the frontier are neither $\triangleright$-complete nor $\triangleright$-compatible. However,

they offer an important insight into how useful the method of generating APF actually is. Zietzler et all also show that by using a single indicator, or even any finite set of indicators, it is not possible to achieve $\rhd$-completeness and $\rhd$-compatibility at the same time. The best we can achieve is either $\rhd$-complete and $\not\rhd$-compatible or only $\succ\succ$-compatible but with no completeness.

Therefore, in our comparisons we aim at presenting a range of indicators which give a good insight into the quality and practicality of the frontier generated. In comparison and evaluation of the method for the generation of APFs we will use I^{NDHVI}, I^{time}, $I^{Avg(R)}$, $I^{Var(R)}$, $I^{\#-plans}$. We believe that this mix of indicators presents different aspects of the frontier and gives the user a meaningful insight into which methods best suit their needs. It also allows us to qualitatively compare different systems.

5.4 APF generation approach

5.4.1 Introduction

Section 5.1 presented some techniques used for generating frontiers of plans. In this section we present a method for the generation of APFs of plans. It is based on a well known method in operations research of combining metrics into a single one using a weighted sum. This single objective is then used to constrain the solution space and calculate a solution on the frontier. In planning this process cannot be applied directly. It is not possible to specify, based on the plan metric, what the resource consumption of a plan is going to be. However, in certain conditions, it is possible to use the plan metric to guide a search towards areas of the search space likely to contain a plan which populates a specific area of the frontier. By repeatedly directing a planner into different areas of the metric space, it is possible to obtain an APF of plans. We start by discussing the most popular, weighted sum of metrics, approach to populating the frontier, its different variants, strengths and weaknesses.

5.4.2 Weighted sum approach

The weighted sum of plan metrics (WSPM) approach to generation of a frontier of plans is a method for combining different plan metrics, used to evaluate plans, into one plan metric using a weighted sum. A planner is then used to find a plan optimising this function and to therefore obtain a plan on an APF. Repeating these steps, using a selection of weightings, gives a populated APF of plans. The selection of weights to apply to different plan metrics varies between approaches. After a short discussion of similarities between planning and OR approaches, this section will present different methods that could be used in planning.

In OR, a weighted sum approach is very popular. It is used, in its different versions, by many techniques for calculating the APF such as Physical Programming or Normal Boundary Intersection (described in Section 5.1.4). Where in PP [42] an algorithm is given to calculate weights used to combine different metrics to generate an even distribution of points across, also concave, frontier. As explained [11], depending on whether the frontier is convex or concave, it is in many cases hard or impossible for a weighted sum approach to generate an even distribution of points. In many cases large areas of concave frontiers are left under-represented.

It is important to note the difference between the effects of applying the WSPM method in OR and planning. In OR after weighting different metrics, an algorithm is applied to calculate the solution lying on the APF. Depending on the algorithm, points on an actual PF can be generated. However, in planning, creating a problem with a weighted function does not mean a planner will generate a plan in the expected area of the solution space. In fact, as shown in Section 3.4 metric sensitivity is a very rare property for a state-of-the-art planner. This is an important motivation for introducing the metric sensitive property for planners.

5.4.3 Calculating weight vector

Planning metrics can be combined into a single function in a variety of ways. The one which we use most often generates weights based on a solution to a simplex problem. For N metric functions we will generate the weights α_i by solving $\sum_i^N \alpha_i = 1$. There is an infinite number of solutions when $\alpha_i \in \mathbb{R}$. To limit the number of solutions we impose bounds on each α_i. For example for $\alpha_i \in \{0, 0.1, 0.2, 0.3, ...0.9, 1\}$ we obtain a limited number of solutions.

The weights can also be distributed non-uniformly. This approach might be useful when, for example, we want to explore one area of the search space more than the others. This could be due to expecting to find a higher density of solutions in this area.

5.4.3.1 Benefits

The weighted sum approach gives us many benefits. Its main benefit is the ease of use: the only requirement is the vector of weights. The work associated with finding appropriate plans is outsourced to a solver or a planner. It also gives us an easy mechanism for calculating weights and deciding which areas of the search space to focus on. In comparison with many other methods, calculating weights is computationally cheap.

5.4.4 Improving the distribution (ED strategy)

The weights can also be calculated to reflect the expected properties of the APF. We have developed a method of iteratively calculating weights based on the current partial-APF and a distribution measure from Section 5.3.1 defined in Definition 5.11. The aim of this method is to obtain an even distribution (ED) of points on the APF. A weight vector $\overrightarrow{\alpha}_i$ based on a partial APF_{i-1} is calculated by finding the most underpopulated area on the APF, expressed as the largest spanning circle, as described in the algorithm for distribution represented in Figure 5.10 on Page 112. The centre of such a circle is used as the point of attraction. This algorithm

aims at generating points in under-represented areas of the search space and achieves a better distribution of plans in the APF.

For the largest spanning hypersphere H, with coordinates $[h_1,...,h_N]$, the vector of weights which attracts search to the centre of this circle is: V = $[v_1,...,v_N]$ where $v_i = 1 - h_i/max(H)$, and max(H) is the largest of h_i. In other words, first, each of the coordinates is scaled to a number from the range [0..1], then this number is subtracted from 1. The reasoning behind this subtraction is that if a plan where the search should be directed has a high value of a particular resource, the cost of this resource should be low. The weights in the vector represent the costs of using a unit of a metric. Therefore, a plan with a scaled amount of used resource of 0.9 is being sought for using a weight of 0.1 on this resource. Conversely, if a resource should not be used, and its scaled usage is 0, the weight on that resource is 1 which makes it an expensive resource and a metric sensitive planner should restrain from using this resource.

This algorithm is called the even distribution (ED) strategy.

5.4.5 Selecting an appropriate planner

To generate solutions in the areas of the search space indicated by the weight vector, a metric sensitive planner is required. If a planner is not metric sensitive, it ignores the metric function and therefore, no matter which weighting we provide, the resulting plan will be the same. Interesting cost trade-offs leading to richer shapes of pareto frontiers are present in domains with state dependent action cost. As shown in Figure 3.2 in Section 3.4.2.1 few planners behaves in a metric sensitive way in such domains. Our choice is therefore limited to metric sensitive planners. Later in this chapter, we answer the question of how metric sensitivity impacts on the performance of planners in generation of APFs. Based on the experiment results from Section 4.3, the following planners are used: LPG and POPF2 with compilation.

5.4.6 Conclusion

We have described work in the OR community related to generating APF, and using the weighted sum approach. The weighted sum approach is a simple, light in calculation, approach to generating APF. It can be applied in both OR and planning. However, the details of the algorithm and its interpretation are different in both fields. A selection of approaches to generating vectors of weights for the generation of APF was presented and briefly discussed. Further, in this work, when we refer to the weighted sum approach we will mean the simplex solution of weights and denote it the weighted sum of plan metrics (WSPM), unless explicitly stated. This approach in combination with use of a metric sensitive planner should give us an effective way of generating good quality APFs.

5.5 Improved APF for stochastic planners

Stochastic planners typically generate a different frontier every time they are run. Although that might generate a poorer quality solution, it is a very interesting property. This section describes a method for exploiting the stochastic behaviour of planners to further improve the quality of the frontier. We start by describing operations for merging APFs and then describe the method and its evaluation.

Merging two frontiers is an operation where an APF is obtained from merging sets of solutions from two or more APFs. This requires removing duplicates and dominated solutions from the resulting set. We denote this operation as $\phi = \phi_1 \oplus \phi_2$, as stated in Definition 5.2.

Following the proof in Theorem 5.9, the resulting frontier ϕ is of no worse quality than any of the composite frontiers. Therefore, by merging different APFs, it is possible, and often the case, to obtain a higher quality frontier.

This process can be repeated iteratively and for each iteration an APF ϕ^i is obtained. The improvement on the frontier is bound by the optimal PF. In the next section, we show that it is

also limited by the properties of a planner. Therefore, for large i, ϕ^i is either going to converge to a value, specific for the planner when the planner cannot generate higher quality solutions, or to the optimal pareto frontier. We call the frontier to which merging multiple APFs converges as the final approximation of pareto frontier.

Definition 5.20. Final Approximation of Pareto Frontier (FAPF), for a sequence of APFs ϕ_i is the $\varinjlim_{i\to\infty} \phi^i$ where ϕ^i is:

$\phi^0 = \phi_1$

$\phi^i = \phi^{i-1} \oplus \phi_i$

Section 5.7.4 contains an examination of the FAPF and the values it converges to for various planners and domains.

5.6 MOPS

The Multi-Objective Planning System (MOPS) was designed and created to effectively plan for multi-objective domains. It handles the extended PDDL 2.1 domains with multiple plan metrics. The output of the system is a frontier of plans. MOPS can work with different weighted sum strategies, such as the ones described in Section 5.4 or any other, more complex, strategy designed by its user. It internally uses a metric sensitive planner to solve problems with single metric.

The user can determine which plan metrics are relevant to them, what strategies they want to use, as well as, which metric sensitive planner should be used. That gives the user the power to shift between the speed, quality, distribution and number of plans required in the APF. All of the components of MOPS are described in the coming sections starting with a general architecture, followed by pre-defined strategies, planners and its various configurations.

5.6.1 MOPS design

MOPS is designed with flexibility in mind. Its main feature is the ability to allow a user to easily plug-in any new metric sensitive planner or a user defined strategy. Figure 5.12 shows a simplified UML class diagram for MOPS. New planners and strategies can be easily implemented following a defined interface and can then be plugged into the system. This allows anybody to easily try their planner within the framework.

Steps taken by MOPS to solve a multi-objective planning problem are as follows:

- Parse domain and problem file.
- Using a strategy, generate a set of domains and problems with a single plan metric.
- Use a planner to solve these new planning problems.
- Evaluate the solutions, treating each of the original plan metrics as a dimension.
- Construct an APF from the set of generated plans.

Experiments are reported below with the weighted sum strategy executed. We assume that there is a relatively small number of weights and, therefore, problems to solve. In practice, the number of weights is usually under 100.

5.6.2 MOPS strategies

A strategy, as previously mentioned, is an interface for a generic experiment strategy. It provides a single method which, for a given problem and a domain, returns a set of problem instances to solve. Conceptually, a strategy is a way of using the planner to obtain the APF. For a given input domain and a problem file with multiple metrics, a strategy is responsible for generating a set of domains and problem instances which, if solved with a metric sensitive planner, generate a set of plans populating the metric space. The way that the metric space is

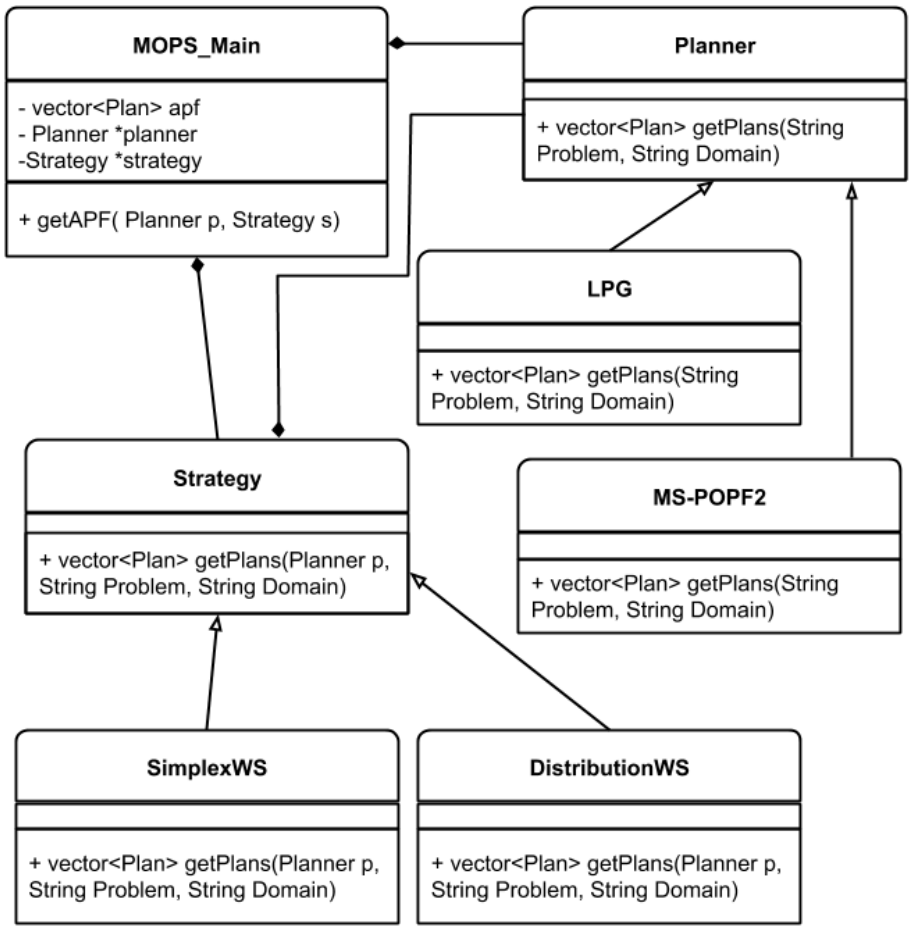


FIGURE 5.12: Simplified UML class diagram which exhibits the main elements of the architecture. It shows how the components of MOPS, including Strategies and Planners, interact.

populated depends on the intention of the strategy. For the purpose of this research the most common use of a strategy is to aim at generating a well populated approximation of a pareto frontier. Currently implemented strategies include:

1) Simplex based, weighted sum of plan metrics

This strategy uses a simplex method to pre-calculate weights for plan metrics. Once all weights

are known, it generates multiple problem instances with different weightings on the plan metrics. For example, one of the simplices for a problem with two plan metrics which we use is: $\alpha_1 + \alpha_2 = 1$ for all $\alpha_i \in \{0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1\}$. This yields the following 11 pairs of weights on the metrics: {(0, 1) ,(0.1, 0.9), (0.2, 0.8), (0.3, 0.7), (0.4, 0.6), (0.5, 0.5), (0.6, 0.4), (0.7, 0.3), (0.8, 0.2), (0.9, 0.1), (1, 0)}. The same technique is used for the three and more dimensional problems. The simplex used for N dimensions is:
$\sum_{i=1}^{N} \alpha_i = 1, \alpha_i \in \{0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1\}$.
This strategy is denoted as the Weighted Sum of Plan Metrics (WSPM).

2) Even distribution

The goal of this strategy is maximising even distribution across the entire APF. As a means of achieving this, it uses the spanning hyper-spheres from each of the APF points as described in Section 5.3.1. Then weights are calculated to lead the planners towards the middle of the largest hyper-sphere. This is achieved by using the formula presented in Section 5.4.4. This strategy is denoted as the Even Distribution (ED).

3) Rerun a planner This strategy was developed to test how well stochastic planners perform using only stochasticity, and without any knowledge about the metrics. The planner is re-run multiple times. However, neither the weights on plan metrics nor the plan metrics themselves are present in the problem description. When comparing with other methods, a planner is re-run a number of times corresponding to the number of executions within the method for comparison. For example, if a simplex method is compared where 66 different weights are used, a planner would also be run 66 times. This strategy is denoted as the Multiple Re-Runs with No Plan Metrics (MRNPM).

5.6.3 MOPS planners

MOPS is designed for reuse and extensibility. It defines an interface for a planner which allows any planner to be easily used within the framework. All planners that are used in MOPS must

implement the Planner interface which contains one method:

```
class Planner
{
public:
  virtual std::vector< std::string, std::allocator< std::string > >
    P_GetPlan(
      std::string s_problemInPDDL,
      std::string s_domainFileName);
};
```

This method takes a plan and a problem description in PDDL as input, and returns a vector of plans in text format. Actions in each plan are separated by a new line.

To make the use of MOPS even easier, it loads planners as dynamic libraries and there is no need to recompile MOPS as a system. This allows planners to be plugged in quickly and efficiently into MOPS. A simple way of making a planner discoverable for MOPS is described in the documentation. The documentation is attached as an Appendix 7.2

5.6.4 MOPS configurations

In order to facilitate efficient comparison between planners and strategies the following planners have been tested: LPRPG, LPRPG-stochastic, POPF2, POPF2-stochastic, POPF2-compilation, LPG with option -n 1, -n 2, and -n3 in settings as "deterministic" and stochastic, also with the compilation method described in Section 4.2. The following strategies, discussed in the previous section, are also available: WSPM, ED, MRNPM.

The first two strategies can be used with any planner and their effectiveness is assessed in the experiment result section. The multiple-re-run strategy can only be used with stochastic planners. It offers a sense of how much stochasticity contributes to the overall performance of a planner. If used with a deterministic planner, this strategy would always generate the same solution for a given problem. This is not an interesting behaviour within the context of generation of an APF.

5.7 MOPS evaluation

In this section we present results for the evaluation of MOPS and metric sensitive planners within MOPS. Domains for these experiments are selected based on their multi-objective properties. In order to test how well the planners generate pareto frontiers, we need to provide domains which offer rich trade-off between resources and ways of achieving the goal. The metric solution space for these domains is rich in solutions spanning through different areas, representing the trade-offs within the domain. In most current benchmark domains, a single solution usually dominates all others in terms of the cost expressed by the plan metric. This happens due to the strong coupling between plan metrics and the plan length. These domains do not offer scope for an interesting comparison between multi-objective systems.

Experiments are conducted on an Intel®*Core*™ i7-2600 CPU @ 3.40GHz x 8 machine with 4GB RAM memory for each planner.

We first present experiments and results for the comparison of methods of obtaining APFs of plans. Then we go on to discuss specific properties of planners and how they impact on the quality of the frontiers generated. First we show the impact of metric sensitivity, by testing how planners perform with and without the new compilation from cost to temporal domains. Then we go on to discuss the impact of metric sensitivity on the resulting frontiers. Finally, we conclude with an example of how stochasticity can be used to further improve the quality of the frontiers generated.

5.7.1 Qualitative comparison of PF generators

The first series of experiments aims at evaluating how different planners, given their metric sensitive evaluation, perform in generating frontiers of plans. In order to test a planner, MOPS takes as input a multi-objective domain and, using a strategy and the planner, which are its parameters, generates a frontier. Internally, as we described in the previous section, this involves

running the planner multiple times on the same problem instance with different weights on the plan metrics. After merging all plans, and removing dominated plans, a frontier of plans is formed. This frontier is then presented for evaluation using methods described in Section 5.3.

For the purpose of this comparison we use the following planners: LPG in configuration with option -n 1, 2, and 3, POPF2 with compilation from cost to temporal domains, described in Section 4.2. For a fair comparison of all planners WSPM, a simplex based weighted sum of plan metrics strategy, is used with each of them. The strategy generates 66 weights for a three dimensional problems. Therefore, in order to generate a frontier, 66 problem instances are to be solved. Each planner is given 100s to generate a solution to each of the sub-problems, and therefore the total time taken to generate the frontier is limited to the maximum of 6600 seconds (110min). In practice, each planner terminates quicker. Time of generation of the frontiers is presented in Figure 5.14.

The frontiers are evaluated using the following indicators $I^{NDHVI}, I^{TIME}, I^{\#-PLANS}, I^{Avg(R)}$ and $I^{Var(R)}$. This set of indicators covers various aspects of the generation of frontiers such as their quality, time taken to generate, distribution and coverage of the space. $I^{\#-PLANS}$, representing the number of plans in the frontier, together with $I^{Avg(R)}$ and $I^{Var(R)}$, is a proxy for coverage. The number of plans, the variance and average radious of spanning circles, can be used instead of coverage. For two frontiers with the same $I^{Avg(R)}$, the one with more plans covers a larger area of the space and the one with fewer plans covers a smaller amount of the space. That is because, if plans are equally distant from each other, then more plans must cover a larger area of the space.

Since POPF2 with compilation performed so well in the metric sensitivity experiments, we expect it to generate high quality frontiers. The higher metric sensitivity of POPF2 means that it is more responsive to the change of the metrics. Therefore, it can be more precisely directed to look for plans in specific areas of the metric space.

Experiment The experiments were carried out using the MOPS framework with the settings mentioned above. Variance and standard deviation of these results are presented in Appendix 7.4.

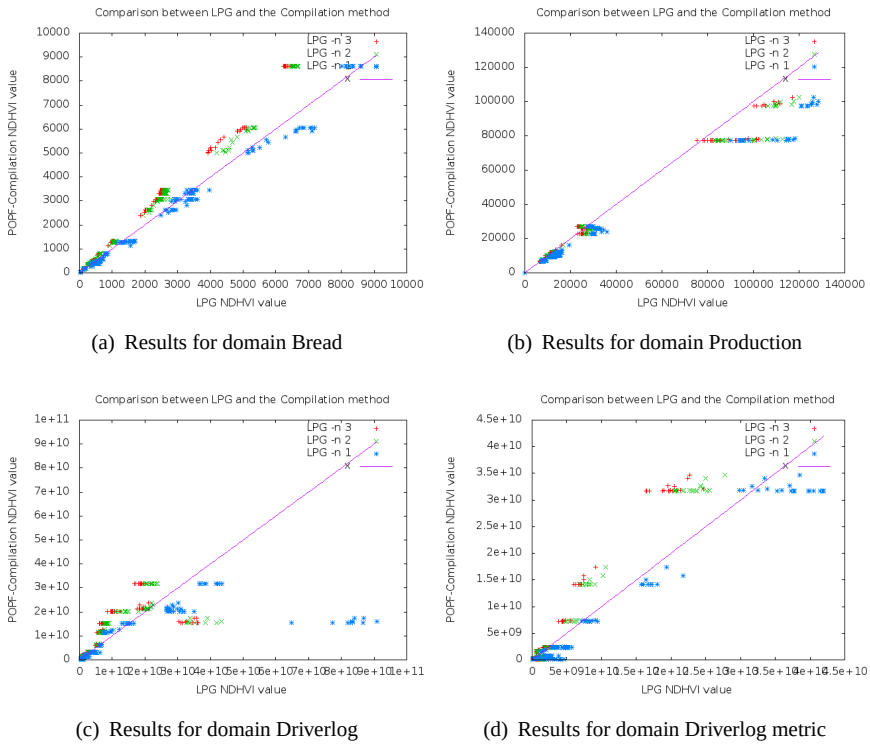
After all results are available, the calculation phase starts. For each problem file for each domain, a reference frontier and a nadir point are calculated. The reference frontier is a frontier resulting from merging all plans found by all planners for the problem. After removing all dominated plans, we obtain an APF with a quality, expressed as I^{NDHVI}, no worse than any other frontier obtained in a single run of MOPS. This reference frontier is referred to as the final approximation of the pareto frontier (FAPF), Definition 5.20. It is later used to calculate the relative quality of frontiers.

The nadir point is a plan dominated by all plans found in all of the experiments for the problem. It is selected such that it is the best, in terms of I^{NDHVI} value, among all plans dominated by all plans found. This plan is used as a reference when calculating the I^{NDHVI} for all frontiers for the problem.

Results The results for I^{NDHVI} are presented in Figure 5.13, I^{TIME} in Figure 5.14, $I^{\#-PLANS}$ in Figure 5.15, $I^{Avg(R)}$ in Figure 5.16, $I^{Var(R)}$ in Figure 5.17. A discussion of each of these indicators is presented below. The section is concluded by a discussion of all of the indicators as a whole.

NDHVI As expected, allowing LPG to take more time and improve on the solution yields higher quality frontiers. This can be seen by the shift from right to left as the "n" parameter increases from 1 to 3.

Although we expected to see a higher quality (smaller values of I^{NDHVI}) for the cost-oriented approach of POPF2 with the compilation to temporal domains, the results for I^{NDHVI} in Figure 5.13 show that this is not always the case. When comparing frontiers formed out of the first plans generated by both POPF2 and LPG -n 1, the quality of POPF2 is higher across all domains.



(a) Results for domain Bread

(b) Results for domain Production

(c) Results for domain Driverlog

(d) Results for domain Driverlog metric

FIGURE 5.13: Results showing the difference in I^{NDHVI} between APF generated by POPF2 with Compilation and LPG in its different configurations. Aggregated numerical values resented in this table are also available in Appendix 7.4. Lower values are better.

When LPG is given more time, it can produce better plans, which result in a frontier of higher quality than the one generated by POPF2. One additional factor which contributes to such a big increase in the plan quality and the frontier quality generated by LPG is the fact that when used with the -n 2 option it generates two plans for every problem and with the -n 3 option it generates three plans for every problem. This effectively creates an APF from 132 and 198 plans for the options -n 2 and -n 3 respectively. As we will show later, variety of solutions impacts on the quality of the APF.

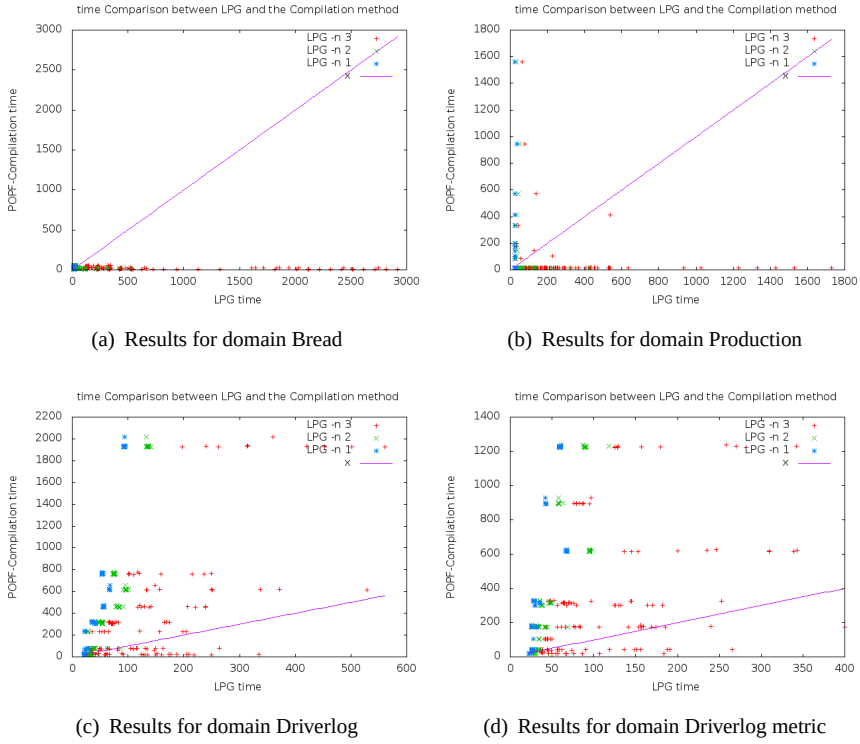
What is also clear after examining the images in detail is that, when the experiment is repeated multiple times, LPG generates different frontiers, of different quality. This creates a relatively

large spectrum of qualities generated for a single problem. At the same time POPF2 consistently generates frontiers of equal quality, which results in horizontal bars in the Figure 5.13.

Numerical results for these experiments are available in Appendix D, Section 7.4.

Time Figure 5.14 shows the difference in time it takes for each of the systems to generate the frontier. Since the total time allowed for each plan is 100 seconds, and there are 66 problems solved for each frontier, the maximum total time a planner could take is 6600seconds. The figure suggests that planners took much less time than they were allowed to. However, examining each of the plan executions separately reveals that some weightings of the plan metrics are harder to solve for a planner than others. Therefore, a high value for the total time usually indicates that some of the plans were not generated and the planner timed out. For LPG with the -n option higher than 1, that usually means a poorer quality plan is generated. However, for POPF2 this usually means that no plan is produced, and therefore the frontier might be missing some good solutions. This is one of the reasons why the quality of the frontier generated by POPF2 sometimes differs between two runs, although it is a deterministic planner. On average, POPF2 takes more time than LPG. All aggregated results are available in Appendix 7.4, and show more accurately what the difference between these planners is.

Cardinality of APF The number of plans forming the APF, $I^{\#-PLANS}$, as represented in Figure 5.15, shows that the configuration of LPG which generates more plans (-n 2, -n 3), also generates larger frontiers. When looking at these results it is important to keep in mind that POPF2 and LPG -n 1 generated a maximum of 66 plans, whereas the configurations of LPG with -n 2 and -n 3 generated 132 and 198 respectively. A value of 20 means that 20 out of 66, 132 or 198 plans were non-dominated and form the frontier. Frontiers with a higher cardinality usually cover more areas of the metric space and therefore, if the quality is comparable, offer better approximations of the PF. This makes higher values of $I^{\#-PLANS}$ more desirable. In order to make sure that there are no large clusters in one part of the metric space and a



(a) Results for domain Bread

(b) Results for domain Production

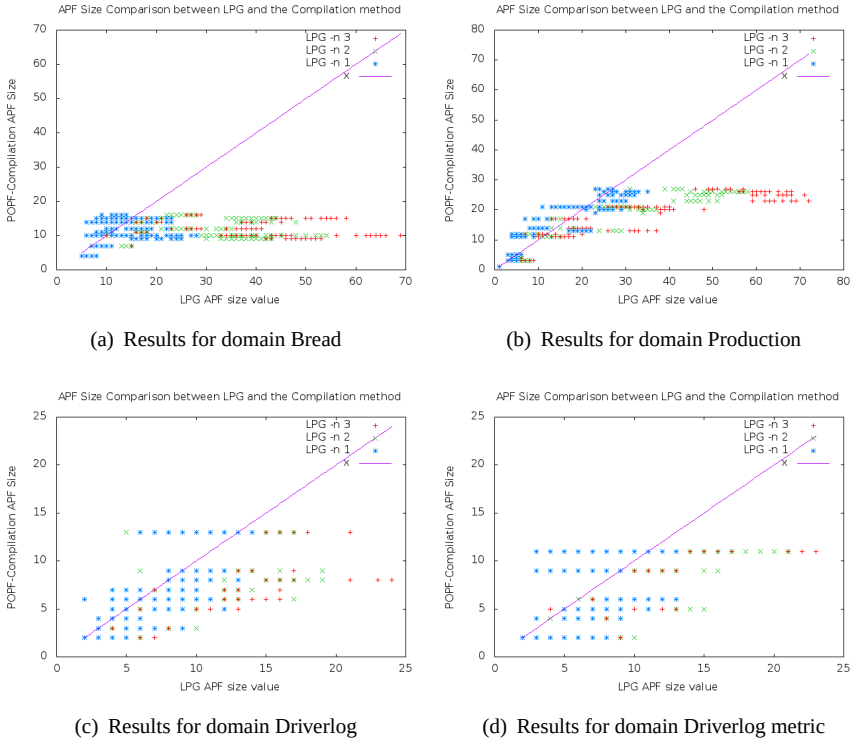
(c) Results for domain Driverlog

(d) Results for domain Driverlog metric

FIGURE 5.14: Results showing the difference in I^{time} between APF generated by POPF2 with Compilation and LPG in its different configurations. Aggregated numerical values resented in this table are also available in Appendix 7.4. Lower values are better.

lot of under-populated areas other metrics must be considered. One of these metrics, average distance between plans on the frontier, is described below.

Average and variance of the radius of the spanning hypersphere Results for $I^{Avg(R)}$ are presented in Figure 5.16. This measure tells us how far from each other are the plans forming an APF. It is complemented by the variance of these radiuses, $I^{Var(R)}$, shown in Figure 5.17. Both of them together give us a good notion of how dense the plans are on the frontier. The distance is incomparable between different problems or domains. For some domains, Bread and Production in Figure 5.16 a,b), the distances are much smaller than for the Driverlog domain in Figure 5.16 c,d). This does not mean that frontiers for Driverlog are more sparse. The reason for such a big difference is the distance of all plans from the point $\vec{0}$. For Driverlog domain,



(a) Results for domain Bread

(b) Results for domain Production

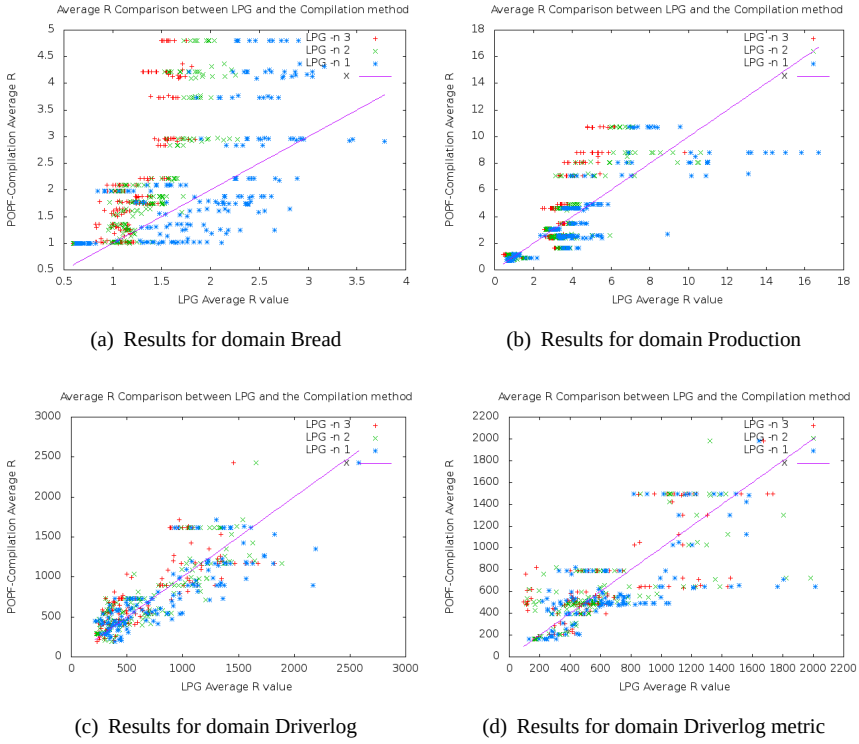
(c) Results for domain Driverlog

(d) Results for domain Driverlog metric

FIGURE 5.15: Results showing the difference in $I^{\#-plans}$, the number of plans, on the APF between APF generated by POPF2 with Compilation and LPG in its different configurations. Aggregated numerical values resented in this table are also available in Appendix 7.4. Higher values indicates more plans, but does not automatically mean the frontier is better.

all plans have very large values for all resources, which results in larger, nominal, distances between plans.

For two frontiers for the same problem, all else equal, larger $I^{Avg(R)}$ means that the plans are spanned across a larger area of the space. In practice, the larger values of $I^{Avg(R)}$ go in pair with smaller values of $I^{\#-PLANS}$, which implies there are fewer plans on the frontier, and therefore they are further from each other. High variance, $I^{Var(R)}$, is caused by some underpopulated, or overpopulated areas of the metric space. It is usually more beneficial to produce evenly populated frontiers with a low $I^{Var(R)}$ value.



(a) Results for domain Bread

(b) Results for domain Production

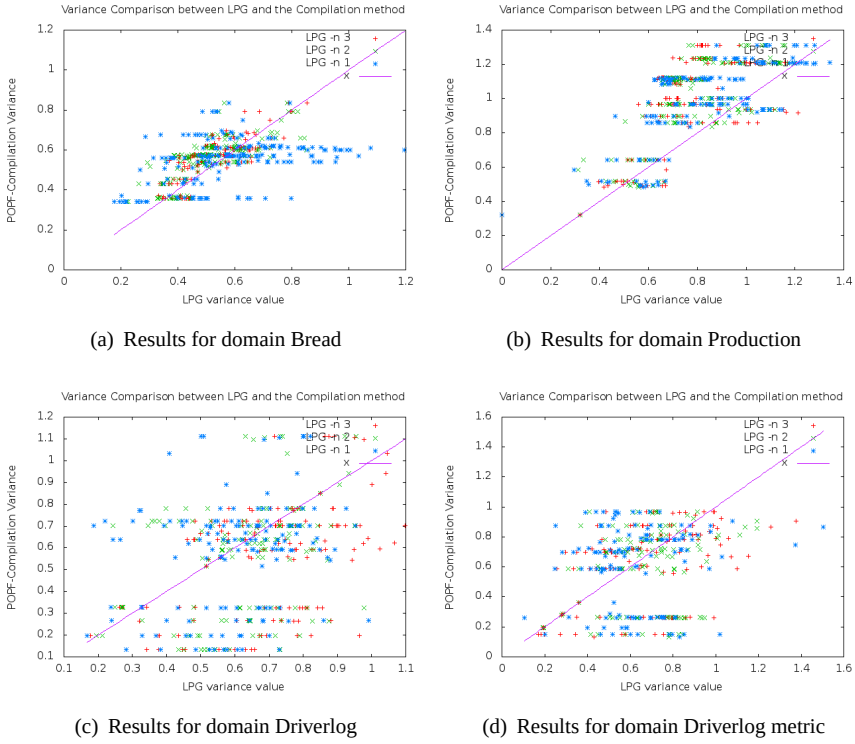
(c) Results for domain Driverlog

(d) Results for domain Driverlog metric

FIGURE 5.16: Results showing the difference in $I^{Avg(R)}$, the average radius of the spanning hypersphere, of APF generated by POPF2 with Compilation and LPG in its different configurations. Aggregated numerical values resented in this table are also available in Appendix 7.4.

5.7.2 Impact of metric sensitivity

With the increase of metric sensitivity of a planner, there comes an increase of the precision with which it is possible to direct the planner into various areas of the metric space. We believe that this precision can help in more precisely targeting specific areas of the metric space and, therefore, generate a well populated APF of high quality. In this section we present experiments where we check how the implementation of the cost-based RPG, Section 4.1, using compilation from cost to temporal domains, Section 4.2, impacts on the quality of the APFs found by both, the POPF2 and LPG planners.



(a) Results for domain Bread

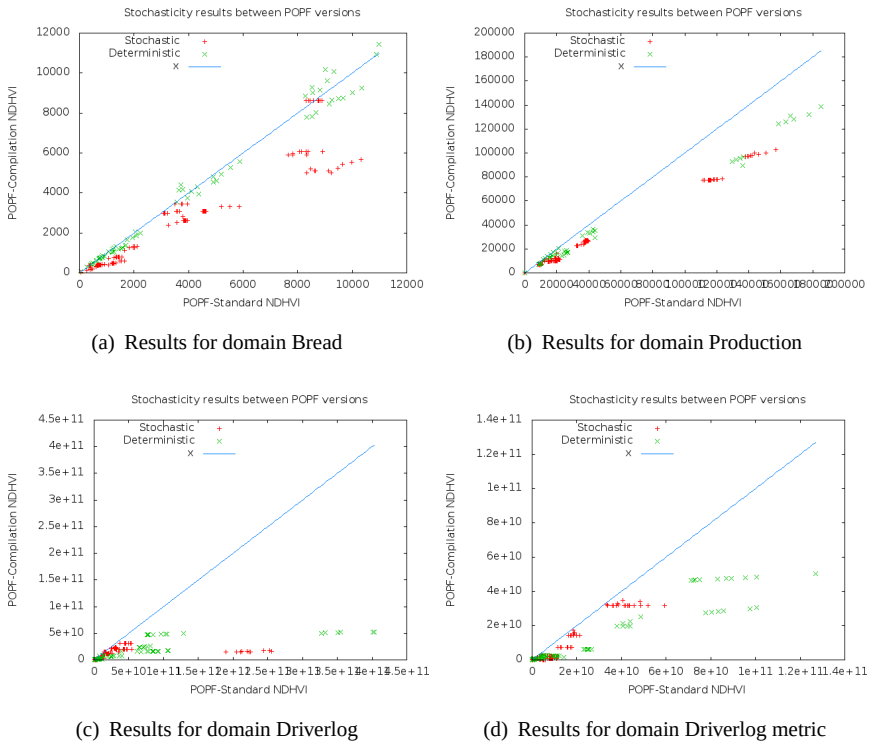
(b) Results for domain Production

(c) Results for domain Driverlog

(d) Results for domain Driverlog metric

FIGURE 5.17: Results showing the difference in $I^{Var(R)}$, the variance of the spanning hypersphere, of the APF between APF generated by POPF2 with Compilation and LPG in its different configurations. Aggregated numerical values resented in this table are also available in Appendix 7.4. Lower values are better.

In order to determine how the compilation method, Section 4.2, impacts on the ability to generate APFs, an experiment with both POPF2 and LPG is carried out in two configurations. The first configuration is the basic version of each of the planners. The second, is the version of the planner with an input being compiled into a temporal domain, as described in Section 4.2. This compilation, in combination with the TRPG used in POPF2, should work as a construct similar to the cost-based RPG as described in Section 4.1 and should increase the metric sensitivity and therefore the quality of solutions. The same compilation is used with LPG. However, with LPG we do not expect to see as high improvement. LPG considers both, the time and cost in its heuristic. Therefore compiling cost into time should make LPG consider the same cost twice while planning, first as the cost and also as the duration of the plan.



(a) Results for domain Bread

(b) Results for domain Production

(c) Results for domain Driverlog

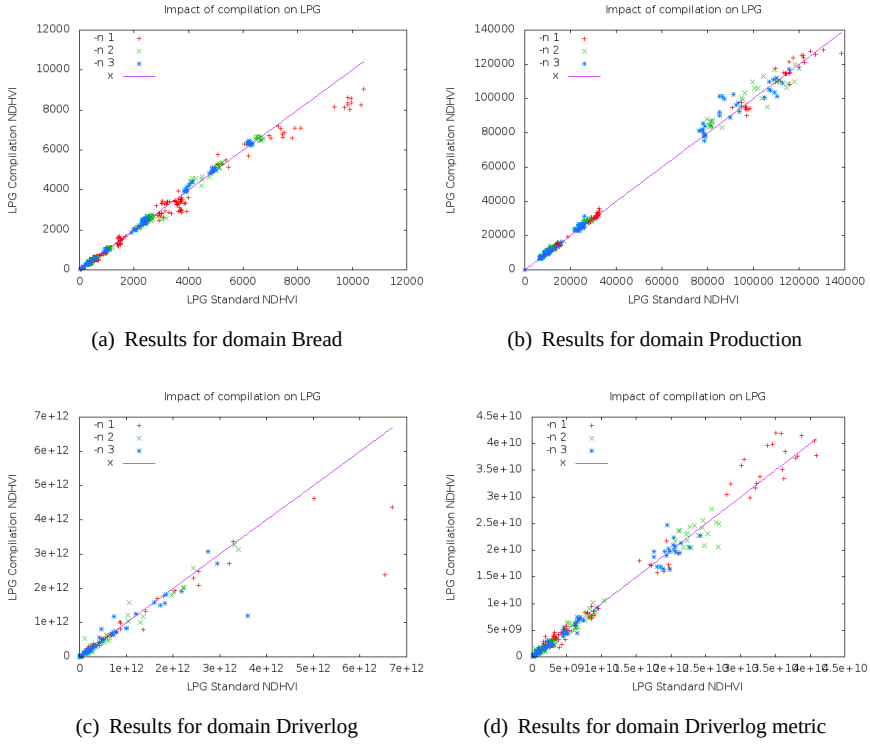
(d) Results for domain Driverlog metric

FIGURE 5.18: Results showing the impact of compilation, from cost to temporal domains described in Section 4.2, on the quality of the APF generated by POPF2 as measured by the I^{NDHVI}. Aggregated numerical values resented in this table are also available in Appendix 7.4.

The experiments are conducted in the same set-up as before. For each domain and problem instance there are 66 new problems created which differ only in the plan metric.

Results As expected, the quality of frontiers generated by POPF2 with compilation is much better across all domains, as shown in Figure 5.18. Compilation clearly improved the performance of POPF2. The results for LPG do not show signs of improvement for the configuration with compilation. This has been anticipated as LPG does not construct the TRPG. The compilation method is designed to exploit the TRPG in order to achieve a higher metric sensitivity.

From the experiments, we can conclude that the compilation from cost to temporal domains successfully exploits the construction of the TRPG as a proxy for the cRPG heuristic. This



(a) Results for domain Bread

(b) Results for domain Production

(c) Results for domain Driverlog

(d) Results for domain Driverlog metric

FIGURE 5.19: Results showing the impact of compilation, from cost to temporal domains described in Section 4.2, on the quality of the APF generated by LPG as measured by the I^{NDHVI}. Aggregated numerical values resented in this table are also available in Appendix 7.4.

significantly enhances POPF2's ability to generate high quality solutions for the purpose of populating a frontier of plans. Therefore, metric sensitivity improves the ability of a planner to generate good quality APFs. This behaviour is more apparent in the results for POPF2 than the results for LPG. There is not much effect of the novel compilation method on LPG's behaviour, which further confirms that the construction of the TRPG as a proxy for the cRPG is a successful method.

5.7.3 Impact of stochasticity

The third set of experiments is designed to explore the impact of stochasticity of a planner on its ability to generate an APF of plans. Two experiments are conducted. The first, where

stochastic and deterministic versions of POPF2 and LPG are contrasted. The second, using a stochastic planner, LPG, executed with and without the knowledge of plan metrics.

The execution of LPG without the knowledge of the plan metrics gives us information about how well the stochastic behaviour performs alone. This is a good basis for comparison between the quality of APFs generated by informed and uninformed search. This tells us how well LPG performs, in generating APF, due to only its stochastic behaviour.

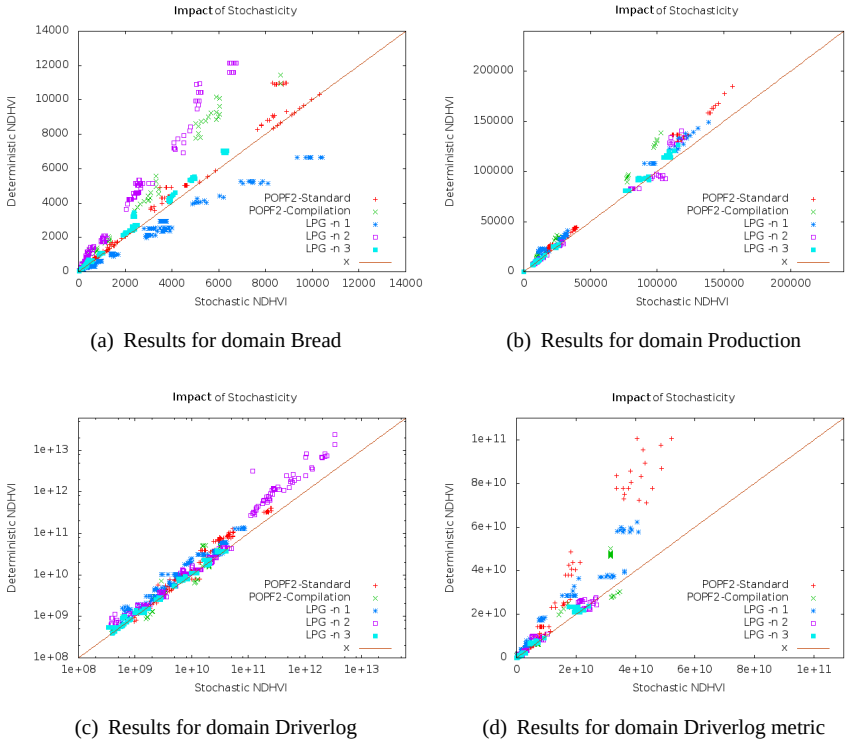
Stochastic vs deterministic First we compare the stochastic version of POPF2 with the deterministic version of POPF2. Stochasticity is introduced to POPF2 in an uninformed way, contrary to stochasticity in LPG. Stochastic behaviour is introduced to change the order of expanding states in the search phase. This does not help in avoiding dead ends, tie-breaking or in any other way. Stochasticity here only allows the planner to randomly discover different areas of the search space, without the consideration for their quality. As expected, this does not lead to better states. However, this technique can greatly improve the quality of the APF by generating a variety of solutions. The second part of this experiment is to run LPG in the same two settings, stochastic and deterministic. Since LPG is naturally stochastic, it is seeded in the same way as described in Section 4.3. This also carries the same pitfalls as described in Section 4.3, but we believe it is a good proxy to show the simple dependency of APF quality on stochasticity.

The set-up for the experiment is the same as for the previous experiments. LPG is used in configuration with option -n 1, 2, and 3, POPF2 with compilation from cost to temporal domains, described in Section 4.2, and in its standard version. The simplex strategy generates 66 weights for a three dimensional problem. Therefore, in order to generate a frontier, 66 problem instances are solved. Each planner is given 100s time to generate a solution to each of the problems, and therefore the total time taken to generate the frontier is limited to the maximum of 6600 seconds (110 Min). After all planners finish the generation phase, the I^{NDHVI} is calculated for each of the frontier.

In Figure 5.20 aggregated results for all planner configurations are shown per domain. Clearly, stochastic configurations perform better than deterministic ones for both planners. This is represented by the majority of results being above the y=x line. These experiments show that stochastic behaviour, enhanced by randomly adjusting heuristic value in POPF2, allows it to generate higher quality frontiers of plans. This is contrary to our findings from the impact of stochastic behaviour on metric sensitivity from Section 4.3. This has an easy explanation, as the frontier of plans benefits from a higher variety of plans. As showed in Theorem 5.9, by adding plans to the frontier we can only improve its quality. Therefore, by finding plans, even randomly, it is possible to increase the quality of an APF. This is not the case when trying to obtain a metric sensitive behaviour. Random steps change the quality of a solution in a non-deterministic way, in some cases improving and in others deteriorating the quality of a single solution. On average that does not carry any benefits. Therefore, although stochasticity is not a useful mechanism for improving the quality of single plans, it is a very desirable property for APF generators.

Stochasticity with no plan metric In this experiment LPG is run with and without the knowledge of the plan metric. We expect LPG to generate a good APF without the knowledge of metrics because its stochastic behaviour causes it to explore different areas of the search space. In this experiment the ability of LPG to find a variety of plans without any knowledge of the plan metric is contrasted with the quality of the frontier it finds when given plan metrics to optimize.

The set-up for the experiment is the same as for the previous experiments. LPG is used in configuration with option -n 1, 2, and 3. The simplex strategy generates 66 weights for a three dimensional problem, for the informed configuration of LPG. Therefore in order to generate a frontier, 66 problem instances are solved and the same number of chances is given to the configuration without any knowledge about plan metrics.



(a) Results for domain Bread

(b) Results for domain Production

(c) Results for domain Driverlog

(d) Results for domain Driverlog metric

FIGURE 5.20: Results showing the impact of stochasticity on the quality, as measured by the I^{NDHVI}, of the APF generated by a planner. Aggregated numerical values resented in this table are also available in Appendix 7.4.

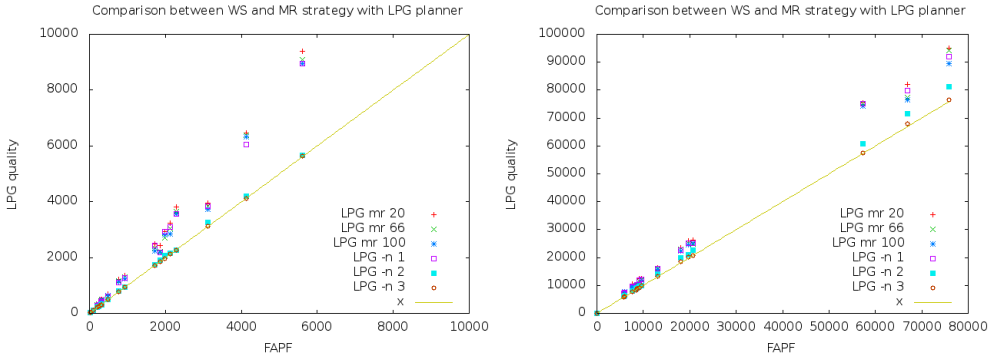


FIGURE 5.21: Results for experiment comparing using multiple re-run strategy with no plan metric knowledge strategy (mr) with the weighted sum strategy (ws) for LPG planner. The results are for domains Bread and Production. The X axis represents the quality of the Final Approximation of the Pareto Frontier (FAPF) as expressed by I^{NDHVI}. This comparison shows the difference between the quality of frontier generated by a strategy and the best frontier found.

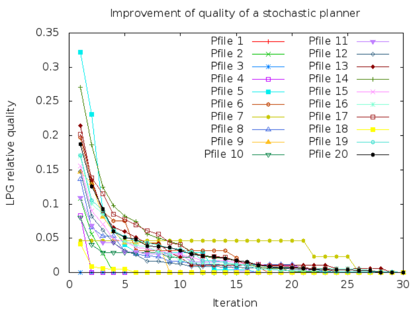
POPF2 is not used in this experiment as it always generates the same plan for a configuration with no metrics. This makes the comparison uninteresting.

Figure 5.21 shows the results for this experiment for domain Bread and Production. Although frontiers generated by the strategy with no knowledge of the plan metric are poorer in quality than the approach based on the weighted sum of plan metrics, the difference is relatively small. Considering the fact that the plans are generated with no knowledge of the plan metrics, and only using stochastic behaviour, it is interesting that these plans can populate a frontier of plans to give a comparable quality with informed metrics. We can conclude that LPG's stochastic behaviour is very beneficial in generating good quality APFs.

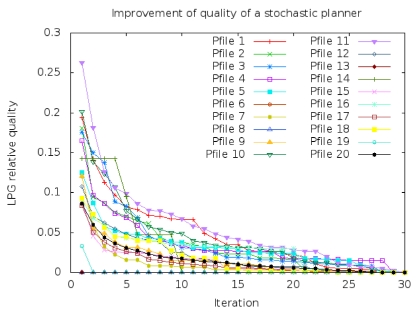
5.7.4 Further improvement in PF quality

Stochastic planners can generate a different frontier every time they are run. The frontiers resulting from different runs can be merged together to form a frontier of higher quality than previous ones. Merging of the frontiers includes removing dominated plans. When merging two frontiers, while using I^{NDHVI} as a measure of quality, the quality can only be increased (smaller values of I^{NDHVI}) (Theorem 5.9). Therefore, by merging APFs we can obtain an APF of a higher quality. We believe that this value for a given planner configuration converges to a value specific to the configuration of the planner. This value is different for each planner configuration and depends on internal properties of the planner.

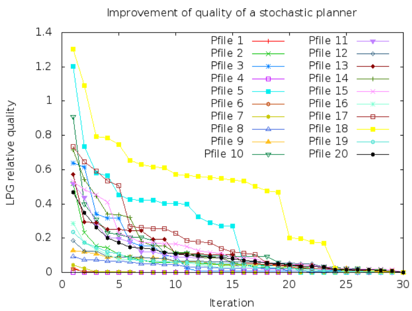
The set-up for the experiment is similar to the previous experiments. LPG is used in configuration with option -n 1, 2, and 3. The simplex strategy generates 66 weights for a three dimensional problem. Therefore, in order to generate a frontier, 66 problem instances are solved. LPG is given 100s time to generate a solution to each of the problems. After all configurations of the planner finish the generation phase, the entire experiment is repeated multiple times. Every time the experiment is repeated, a new iteration of ϕ^i is created, as in Definition 5.20, based on the I^{NDHVI} as the quality measure.



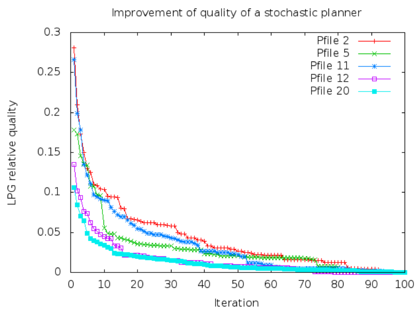
(a) Results for domain Bread



(b) Results for domain Production



(c) Results for domain Driverlog

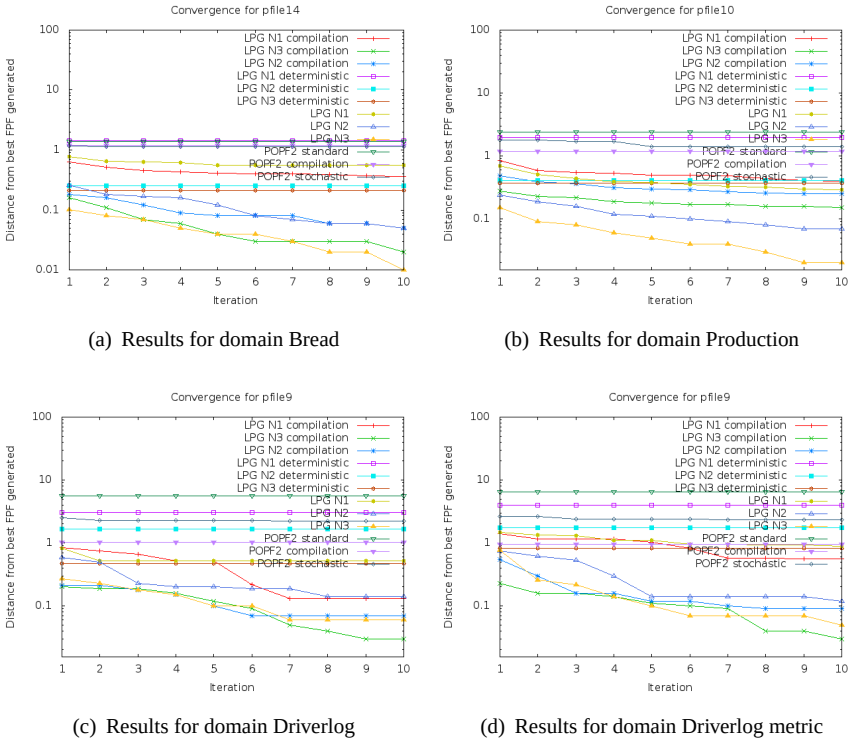


(d) Results for domain Driverlog metric

FIGURE 5.22: Results showing the improvement of an APF after multiple merge iterations as described in Section 5.5.

Figure 5.22 shows results aggregated for LPG over multiple repetitions of the experiment. The three problems used in this experiment are Bread Figure 5.22 a), Production Figure 5.22 b) and d) and Driverlog Figure 5.22 c). Where Figure 5.22 d) presented 100 repetitions of the experiment for domain Production and selected problems. The problems were selected arbitrary out of the ones which, after 30 runs, indicated that they could be improved further.

In all cases the improvement of the quality of the frontier slows down after about 5 to 10 iterations. Every next iteration gives little improvement in the quality of the frontier. The quality after the 10th iteration is usually under 5% from the best quality achieved, after 30 iterations. For the Production domain the improvement between the 10th and 100th iteration for three problems is around 10% which is more significant. However, the effort to generate 90



(a) Results for domain Bread

(b) Results for domain Production

(c) Results for domain Driverlog

(d) Results for domain Driverlog metric

FIGURE 5.23: Results showing the improvement of an APF after multiple merge iterations for selected problems from each domain.

more frontiers is also *very significant*.

In order to show that each method for generating frontiers converges to a different value, results for different configurations of planners are presented together in Figure 5.23. This compilation of results demonstrates two things. First, that only stochastic planners produce the improvement, which is highlighted by the horizontal lines for all deterministic planners; second, that each planner produces quality at a different level.

In conclusion, merging of the APFs of plans is a good way to obtain higher quality frontiers but this carries additional work. The quality of the resulting frontier appears to be limited by the properties of the selected planner.

5.7.5 Even distribution strategy

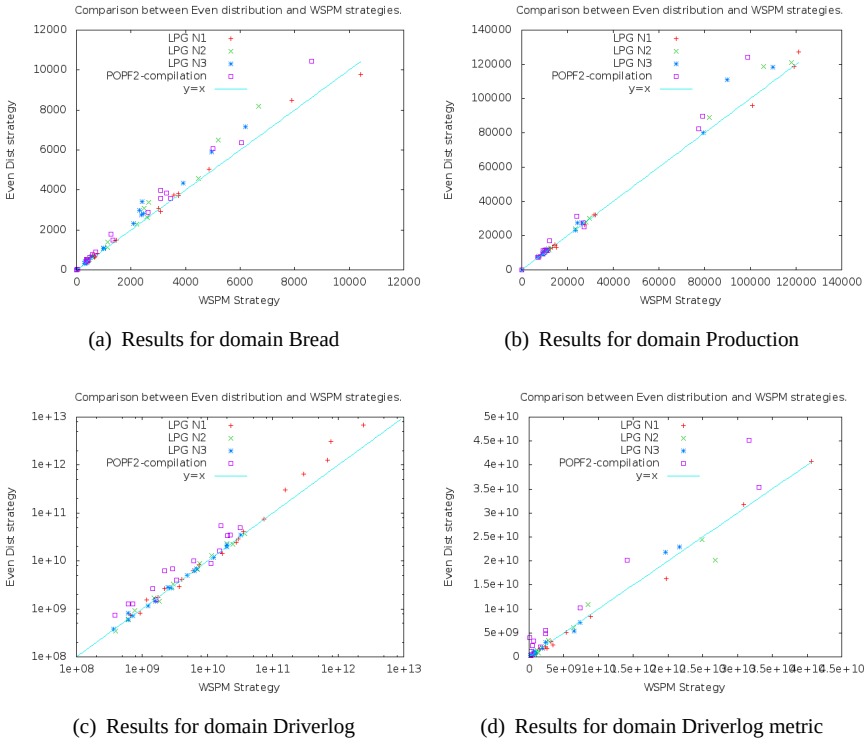
In this section we present a discussion of performance of a method for generating APFs based on the ED strategy. The ED strategy uses weights on metrics calculated based on the algorithm for even distribution presented in Section 5.4.4. The aim of this experiment is to examine what is the impact of directing the search to under populated areas of the solution space. We expect that the resulting frontier should be more evenly populated. This approach can potentially lose some quality in favour of the distribution of plans.

The set up for this experiment is similar to the previously described experiments. Four configurations of planners are used: LPG N1, LPG N2, LPG N3 and POPF2-compilation. Each of the planners is used in the following way:

1. Generate an initial APF by optimizing each of the plan metrics separate.
2. Find the largest spanning hypersphere on the current frontier.
3. Use the centre of the largest spanning hypersphere as a point of attraction.
4. Repeat from 2.

This method aims at producing new plans in the under populated areas of the search space. By repeating this process we expect to eliminate the under populated areas of the solution space.

The results for I^{NDHVI} in Figure 5.24, indicate that the quality of the APF generated by the ED is of poorer quality than the one generated by the WSPM strategy. This can be explained by multiple factors which impact the value of I^{NDHVI}. A first important observation is that the size of the frontier, $I^{\#-plans}$, is significantly smaller for the frontiers generated by the ED strategy. Taking into consideration that it is possible to obtain a higher quality frontiers by merging diverse solutions, as shown in Section 5.7.4, by finding fewer plans the ED strategy is disadvantaged.



(a) Results for domain Bread

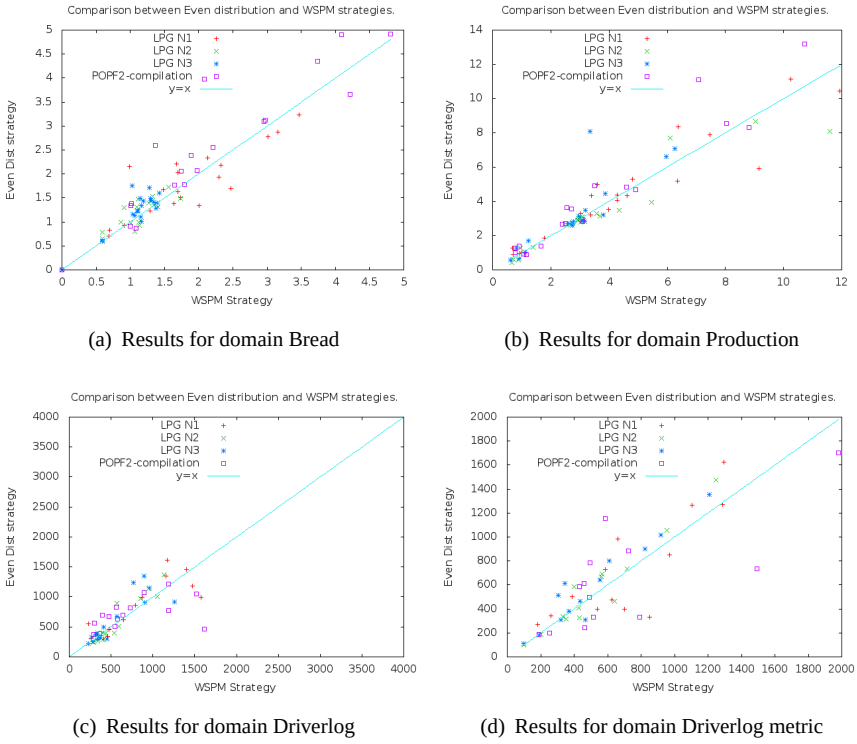
(b) Results for domain Production

(c) Results for domain Driverlog

(d) Results for domain Driverlog metric

FIGURE 5.24: Results showing the difference in I^{NDHVI} between the even distribution strategy and WSPM strategy.

The reason for finding fewer solutions is not as obvious. Both strategies are allowed to run the planner 66 times. The lower number of plans in the frontiers for the ED strategy can be explained by considering what happens during the execution. We have noticed that when a planner is directed to find a plan in an area of the search space with the largest spanning hypersphere, often it is not able to find a plan that would decrease the hypersphere enough to start looking for a plan in a different area of the search space. Therefore, a significant effort is spent on repeatedly eliminating some of the large spanning hypersphere. This has a larger impact on the results for POPF2 than LPG. This happens because POPF2, being deterministic, generates the same plan for the same weights on plan metrics. This means that if it cannot find a plan within the spanning hypersphere, which it is asked to decrease, it keeps searching for it repeatedly. The stochastic behaviour of LPG helps it to avoid these situations, and when re-run



(a) Results for domain Bread

(b) Results for domain Production

(c) Results for domain Driverlog

(d) Results for domain Driverlog metric

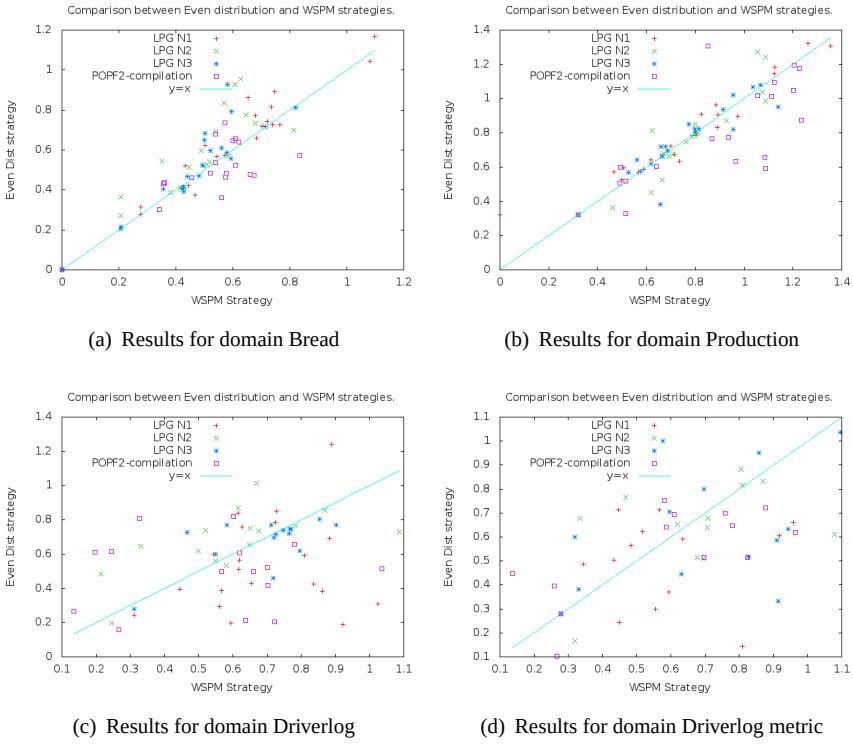
FIGURE 5.25: Results showing the difference in $I^{Avg(R)}$ between the even distribution strategy and WSPM strategy.

with the same metrics it eventually finds a solution reducing the largest spanning hypersphere.

This is also exhibited by a large number of cases where POPF2 timed out and had to be stopped. This is visible in Figure 5.27 for the results for I^{Time}.

The smaller number of plans for ED strategy is also reflected in higher values for $I^{Avg(R)}$. For two frontiers generated for the same problem, plans in the smaller frontier are typically further away from each other.

The only measure which is not in favour of any strategy is the variance of the radius for spanning hyperspheres. The results for $I^{Var(R)}$ are presented in Figure 5.26. It shows that although the domains are of different density, the plans are roughly distributed in a similar way.



(a) Results for domain Bread

(b) Results for domain Production

(c) Results for domain Driverlog

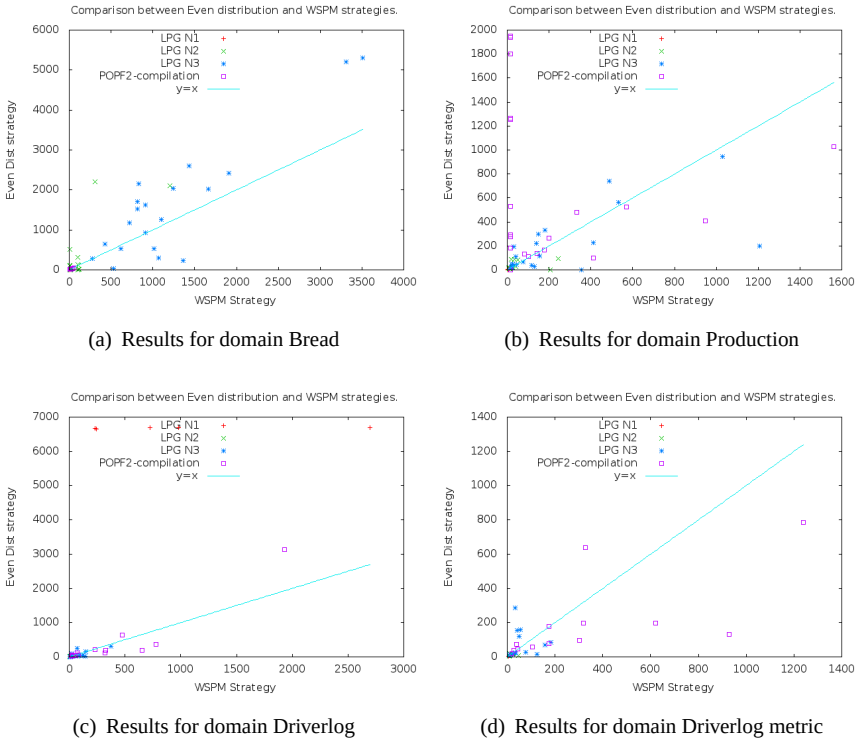
(d) Results for domain Driverlog metric

FIGURE 5.26: Results showing the difference in $I^{Var(R)}$ between the even distribution strategy and WSPM strategy.

This discussion shows how indicators complement each other and give better information about the frontier when considered together. It is clear how the size of the frontier impacts on its quality,I^{NDHVI}, or the average radious of a spanning hypersphere, $I^{Avg(R)}$.

5.8 Summary of the results and conclusion

All of the experiments in Section 5.7 are generated using the Multi-Objective Planning System, MOPS, developed as part of this research. It is a system capable of solving multi-objective planning problems by generating APFs as a means of communicating multiple solutions and the trade-offs between them. It is a very generic system allowing multiple planners and various experiment strategies to be used within a single system. The quality of the plans generated



(a) Results for domain Bread

(b) Results for domain Production

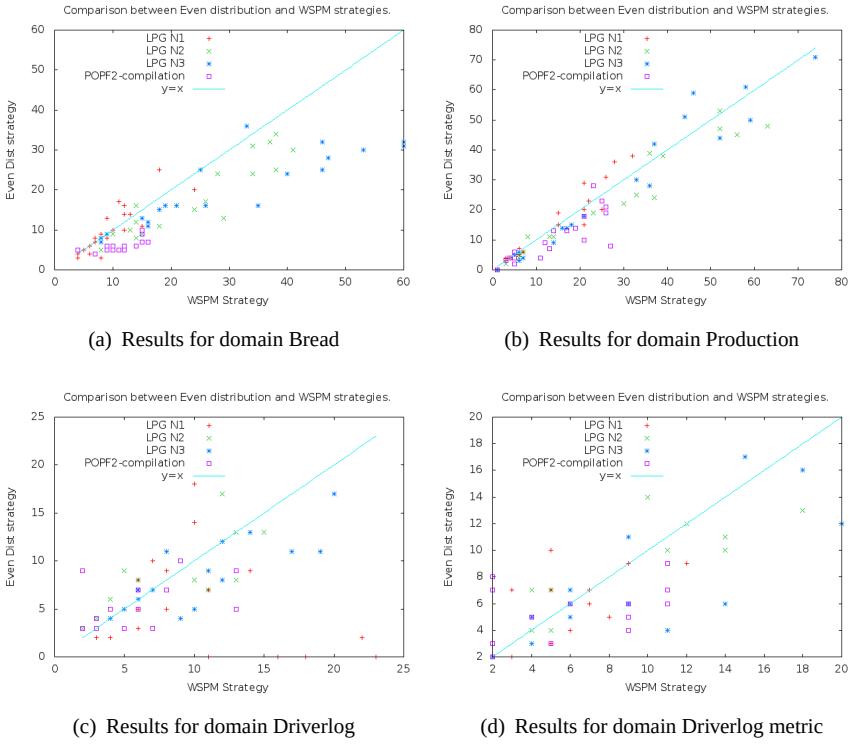
(c) Results for domain Driverlog

(d) Results for domain Driverlog metric

FIGURE 5.27: Results showing the difference in I^{TIME} between the even distribution strategy and WSPM strategy.

depends on both the planner and the strategy used. The experiments have confirmed two major influencing factors on the quality of the APF generated by the system to be metric sensitivity and stochasticity.

The notion of metric sensitivity is introduced in Section 3.2 and Section 4.1 contains a description of a cRPG heuristic which is designed to be metric sensitive. Implementation of this heuristic using a compilation from cost to temporal domains, as described in Section 4.2, and its impact on POPF2 and LPG is examined. The results confirm that the compilation is a successful method of implementing a cRPG. It is also apparent that using the novel compilation method, and therefore adding metric sensitivity to the planner, is a method of rising the quality of the APF which the planner generates, where the quality of the APF is expressed as I^{NDHVI}.



(a) Results for domain Bread

(b) Results for domain Production

(c) Results for domain Driverlog

(d) Results for domain Driverlog metric

FIGURE 5.28: Results showing the difference in $I^{\#-plans}$ between the even distribution strategy and WSPM strategy.

The second factor impacting on the quality of the APFs is stochasticity of the planner. Stochasticity can be used within a planner in multiple ways. It can be a tiebreaker for plateaus where a heuristic function does not provide good guidance, encourage exploration in areas of the search space previously left out by the heuristic function or change the order in which areas of the search space are visited. It can be well integrated into the heuristic function where it allows the planner to explore wider range of areas of the search space, where the quality of these areas is still high, as is the case for LPG. Stochasticity can also allow the planner to explore various areas of the search space, without the concern of the quality, as is the case in the variant of POPF2 described in Section 4.3.3. In Section 5.7.3 both of these cases of stochastic behaviour, result in a higher quality of APFs generated by MOPS in configuration with a planner.

The impact of stochasticity implies the high influence of the variety of results on the quality of APF. This is confirmed in Section 5.7.3 where a completely uninformed stochastic planner, blind to the plan metric, generates APFs of comparable quality to the ones generated by the same planner with plan metric information. Therefore stochasticity alone is a huge contributor to the quality of the APF. One might speculate that really good metric sensitivity would obviate the need for stochasticity. So that stochasticity is a substitute for precise metric sensitivity.

In this section we have determined two of the main factors impacting on the quality of the APF, stochasticity and metric sensitivity. We are able to show that both improve the quality of APF generated. It is also clear that the compilation based implementation of the cRPG improves the quality of APF generated by POPF2. The quality of the frontiers is then of comparable quality with the state-of-the-art planner. All of the results were obtained using MOPS, a system which reasons in multi-objective domains using any of the state-of-the-art planners.

The main contributions of this chapter are:

- Generic system for the generation of frontier of plans.
- Approaches to generation of APF.
- Examination of properties impacting the quality of APF.

Chapter 6

Conclusion

As outlined in the thesis statement, the aim of this thesis is to convince the reader that metric sensitive planners, in combination with plan metric aggregation methods, are an effective tool for the generation of approximations of pareto frontiers of plans. When faced with a planning problem containing context-dependent action costs and plan quality metrics which do not correlate with the plan length, most current state-of-the-art planners ignore the cost completely. In this thesis we introduce the notion of metric sensitivity. It is a property of planners that can deal effectively with domains containing monotonic relations between the plan length and the plan cost. Therefore, it overcomes the common limitation of assuring *strictly correlated* length and cost. An extensive discussion of what metric sensitivity is and how it can be measure is presented in Chapter 3. A novel method for obtaining a metric sensitive heuristic, by creating a cRPG, is shown in Section 4.1. A selection of algorithms is presented, which combine plan metrics into one and use metric sensitive planners, for the generation of diverse plans. Each step of this process is evaluated, metric sensitivity in Section 4.3 and the qualitative evaluation of frontiers of plans in Section 5.7.1. Below, a more detailed summary of each is presented.

6.1 Metric sensitivity

Metric sensitivity is a desirable, but rare, property of current state-of-the-art planners. Metric sensitivity is related to the precision with which it is possible to determine an area of the solution space where the planner will generate a solution. The second factor affecting the precision is stochasticity. A stochastic behaviour allows the planner to produce more diverse plans, at the cost of precision, and usually without a loss of quality.

The notion of metric sensitivity is defined, in Definition 3.1, as the ability of the planner to positively respond to the change of the plan metric. This definition is relaxed to include planners which exhibit the metric sensitive behaviour in Definition 3.4. In Section 3.4 we show that most of the current state-of-the-art planners are not metric sensitive when faced with state dependent action cost models. The only planner that exhibits a metric sensitive behaviour is LPG. Metric sensitivity is a desirable property of planners used for the generation of frontiers of plans. The higher the precision, with which a planner can be directed to specific areas of the solution space, the easier it is to systematically populate the frontier of plans.

One way of attempting to exhibit metric sensitive behaviour is to generate plans stochastically, hoping they are good for some of the metrics. This approach, as shown in Section 3.7, allows metric insensitive planners such as POPF2 to generate different solutions and therefore achieve a degree of metric sensitive behaviour. The results presented in Section 3.7 suggest that stochasticity does not have an impact on the quality of solutions generated. However, the variety of solutions obtained happens at the cost of precision in where the planner finds solutions. This suggests that stochasticity negatively impacts on the metric sensitivity of a planner, while at the same time increasing its ability to generate diverse solutions.

The impact of stochasticity and the metric sensitivity results of current state-of-the-art planners suggest that a deterministic algorithm for generating a plan, while exhibiting a metric sensitive behaviour, is necessary. In Section 4.1.2 we propose a novel, metric sensitive and deterministic,

heuristic based on the cost oriented RPG construction. Optimal planners are, of course, metric sensitive, but are limited to solving a specific class of problems. The heuristic presented in this thesis handles context-dependent action costs.

As a means of implementing the cRPG, a compilation from cost-to-temporal domains is proposed in Section 4.2. This compilation exploits the similarities between the construction of the cRPG and the TPRG constructed by POPF2. Therefore, the compilation uses time as a proxy for cost, the compilation aims at exploiting the TRPG construction in POPF2 and, as shown in Section 4.3.2, there is no effect of the compilation on results generated by LPG, despite the fact that LPG considers action duration in its heuristic.

6.2 Generating PF

One of the applications of metric sensitive planners is the precise generation of plans in different areas of the metric solution space. We discuss the generation, evaluation and presentation of APFs. Although the quality of frontiers of plans is considered, the focus of this research is not on optimality. The generation methods are based on similar approaches to those used in OR as described in Section 5.1.4.

The evaluation of APFs is, in itself, a multi-objective problem. As shown in Section 5.3.3, there does not exist a single fully informative metric for the evaluation of APFs. The evaluation of a set of solutions is meant to help the user to make an informed decision on which frontier generator suits their needs best. Various indicators, used to evaluate frontiers of plans, are presented in Section 5.3.1. Each of the indicators gives information about a different aspect of the frontier. In order to capture a full spectrum of interesting information for the user, a set of indicators must be presented. The indicators proposed in this thesis capture: an overview of the quality of plans contained in the frontier (I^{NDHVI}), the cardinality of the set of plans ($I^{\#PLANS}$), the time it takes for the generator to generate the frontier (I^{TIME}), and the distribution and

coverage of the frontier ($I^{Avg(R)}$ and $I^{Var(R)}$). All of them presented together, allow the user to make a decision on which of the generators is the most suitable for their task.

Currently in planning, there are two main approaches to generating APFs: evolutionary algorithms and methods of aggregation of plan metrics into a single plan metric. Evolutionary algorithms simultaneously generate plans populating the entire frontier. They can naturally *evolve* multiple plans at the same time. The aggregation methods offer more flexibility in the effort spend in searching for plans in particular areas of the metric solution space. They iteratively populate the frontier and, therefore, allow the user to obtain some plans on the frontier at any time.

For the purpose of generating an APF of plans, we present the Multi-Objective Planning System (MOPS). It is capable of generating a frontier of plans using a method of aggregation of plan metrics selected by its user. The strategies for the aggregation used in this thesis are the Weighted Sum of Plan Metrics (WSPM), Multiple Re-runs Without Plan Metrics (MRWPM) and Even Distribution (ED). The focus of each is to exploit a different property of a planner. The WSPM and the ED strategies depend on the ability of a planner to precisely generate a solution in a specified area of the metric space. By repeatedly executing the planner and systematically selecting an even distribution of focus points, the WSPM strategy aims at generating an even distribution of plans across the frontier. The ED strategy follows the measure of distribution presented in Section 5.3.1, and iteratively attempts to improve this metric by focusing the planner on finding solutions in under populated areas of the metric solution space. In contrast, the MRWPM strategy does not use any knowledge of the metric. It is designed to test how a stochastic planner performs without the knowledge of a plan metric. It measures the properties of a set of plans generated by a planner for a given problem. Contrasting the results for all of these strategies can give us insight into how stochasticity and metric sensitivity impact on the ability of a planner to generate APFs.

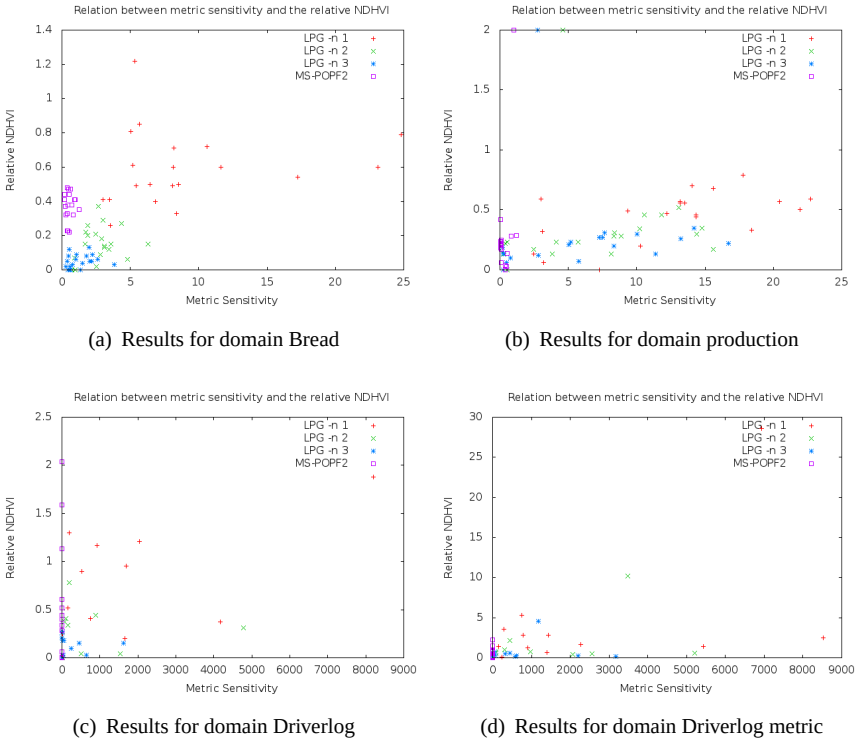
Although a stochastic behaviour allows a planner to generate more variety of plans, as demonstrated in Section 3.7, it has a negligible impact on the quality of the plans produced, as demonstrated in Section 3.7. Therefore, it is interesting to note that, in multi-objective setting, stochasticity improves the quality of the frontiers generated by a planner, as compared with an equivalent deterministic algorithm. This can be observed in results in Section 5.7.3. This brings us back to the question, to what extent can LPG attribute its good performance to its stochastic behaviour? An answer to that question is in the results in Section 5.7.3, where a comparable quality of results is generated for LPG with and without the knowledge of the plan metrics, as represented by the WSPM and MRWPM strategies respectively.

Experiments with the WSPM and the ED strategies exhibit the properties of APF generators based on LPG and POPF2 planners. The results in Section 5.7.2 show that the compilation method of implementing the cRPG allows POPF2 to generate a much better quality APFs. Simultaneously, these results show almost no impact of the compilation on the results for LPG. This is explained, as mentioned previously, by the TRPG construction in POPF2. One might speculate that really good metric sensitivity, providing a high precision in generating plans in the metric solution space, would obviate the need for stochasticity. So that stochasticity is a weak substitute for precise metric sensitivity.

6.3 Summary

Although the two previous sections describe each of the two main focus points of this research separately, namely metric sensitivity and APF generation, it is clear how closely the two go together. An interesting question to answer is to what extent is metric sensitivity correlated with the quality of APF generated? Below we present the last set of results which combines the qualitative results from both sections together.

Correlation between metric sensitivity and NDHVI of APF Here we present an interesting comparison between the quality of plans generated for the experiments for metric sensitivity,



(a) Results for domain Bread

(b) Results for domain production

(c) Results for domain Driverlog

(d) Results for domain Driverlog metric

FIGURE 6.1: Results showing the relation between metric sensitivity of a planner against the relative NDHVI.

presented in Section 4.3, and the results for I^{NDHVI}, as the quality of APF presented in Section 5.7.1, is shown in Section 6.3.

All results from both of the sections are aggregated such that for each domain and problem instance an average metric sensitivity score and average I^{NDHVI}, with relation to the best I^{NDHVI} found, is calculated. The relation to the best I^{NDHVI} found is expressed as $I^{NDHVI}_{score} = \frac{I^{NDHVI} - I^{NDHVI}_{BEST}}{I^{NDHVI}_{BEST}}$ where the I^{NDHVI}_{BEST} refers to the same problem instance as I^{NDHVI}. Set of points for each domain where the x coordinate is the metric sensitivity and the y coordinate is I^{NDHVI}_{score} is presented in Figure 6.1.

There exists a correlation between the high metric sensitivity (low values on the x axis) and high quality of the pareto frontier (low values on the y axis). Contrary to what we expected,

these results suggest that metric sensitivity does not impact the ability to generate high quality APF's as much as some of LPG's properties. We believe that this property is the informed stochastic behaviour. As demonstrated in Section 5.7.3, even without the information about plan metric, LPG generates a good quality APF.

This comparison shows that for a more metric sensitive planner the quality of APF is typically higher (lower values on the Y axis). This correlation is clear when looking at the results for LPG with changing the -n option from 1 to 3. Where for the option -n 3 both metrics, the I^{NDHVI} of the APF and metric sensitivity, are of high quality. Although the improvement of the quality of I^{NDHVI} of the APF is not as big as expected, metric sensitivity is an important factor of generating a high quality APF.

The more metric sensitive the planner is, the higher quality APFs it generates, as is shown in Section 5.7.2. However, as shown in Section 6.3, for two planners, the one with higher metric sensitivity is not guaranteed to generate a better quality of APF. This is mainly attributed to the fact that the quality of APF is influenced by multiple factors, such as distribution of points, and different properties of the planner, such as stochasticity. Based on all of the results we can conclude that metric sensitivity is a desirable property of planners.

Contributions The main contributions of this research are:

1. A definition and exploration of metric sensitivity in planning.
2. A context-dependent, cost-based relaxed planning graph and heuristic.
3. A compilation method from cost to temporal domains.
4. Examination of the impact of planners' properties on the quality of plans and APFs.

In this thesis we have shown that metric sensitivity is an important property of planners. It can be measured based on the relative quality of plans, as measured by the plan metrics. Planners

exhibiting this property are good candidates for APF generators. This is due to the fact that metric sensitivity corresponds with the ability to precisely direct the planner into an interesting areas of the search space. For the purpose of generating an APF, a planner can be directed into each of the interesting areas of the metric space. The results from each of the areas combined give an approximated PF. We have showed that this approach works best if combined with stochasticity. This is due to the fact that variety of plans automatically increases the quality of the frontier.

Chapter 7

Appendixes

7.1 Appendix A. Software implemented for the purpose of this work

7.1.1 Automated compilation from cost to temporal domains

The program for Compilation is implemented in Java. It is built on top of an extended version of PDDL4J [1]. The main extensions which are implemented are the support for multiple metrics in the problem description. After parsing the program performs the translation described in Section 4.2. As an output the program generates two files. First, with a new temporal domain containing temporal and partially grounded actions. Second, which is later used to map the partially grounded actions back to their original names. This second file is a collection of regular expressions which, when applied, automatically parse and change the plan generated for the new domain to turn it into plan solving the original domain.

7.1.2 MOPS

Multi-Objective Planning System (MOPS) is more extensively described in Section 5.6. MOPS is a system which based on user defined strategy generates weighted sum of plan metrics and, using an external metric sensitive planner to solve the single metric tasks, generates approximations of pareto frontier of plans. The system can be used with any metric sensitive planner which complies with an interface defined by MOPS. MOPS can be used to qualitatively compare planners on different domains and different scenarios. Multiple bash scripts that are implemented around it help in conducting multiple experiments and gather data afterwards. Internally MOPS uses a slightly modified version of VAL [2]. VAL is used to calculate the quality of the plans generated in multi-objective domains.

7.1.3 POPF2-stochastic

POPF2-stochastic is a stochastic version of POPF2 [9]. The only difference between POPF2 and POPF2-stochastic is the addition of stochastic behaviour in search. Stochasticity is added in two different levels. First, it randomises branch ordering when expanding the state. Second, in the heuristic evaluation. The first modification reorders actions which are considered equally good in a given state, and therefore solves tie-breaks in a different ways every time it is run and doesn't get stuck in the same places more than once. The second is just an introduction of small noise within the heuristic function which tries expanding actions which would not normally be expanded due to their higher heuristic value. These changes have been implemented for the purpose of testing how stochasticity impacts on metric sensitivity of planners.

7.1.4 LPRPG-stochastic

The changes to LPRPG are the same as to POPF2. Thanks to the fact that LPRPG and POPF2 share the same code base, the same two changes were done.

7.2 Appendix B. MOPS documentation

7.2.1 Introduction

Current planners focus on finding a single solution to a given problem. The decision maker (DM) has a choice of either finding the solution fast using a satisficing planner or finding a high quality solution using an optimal planner. The quality of the solution is evaluated in terms of a metric function given by the DM. That forces the DM to specify preferences in terms of a single metric function which in many real life problems is hard.

This work is based on the need to relax this constraint and to allow the planning process to be carried out without the need to specify the single metric function. A single plan is sufficient when minimising a single objective function, however, it is very unlikely that a single solution would minimize multiple objective functions at the same time. The solution for a problem with multiple metric functions is no longer a single plan but a set of plans which represents the trade-off between different objectives.

Satisficing planners are doing remarkably well in finding solutions to planning problems efficiently. Although they are not optimal, the speed with which they find a solution makes them very valuable. We explore how we can use them to quickly find solutions in various areas of the search space, which combined together give a high quality solutions set.

Our system, the Multi-Objective Planning System (MOPS) is capable of generating sets of qualitatively good plans using different planners within various strategies, developed together with MOPS. MOPS is open source (Distributed under GNU General Public License) easy to extend to use a new planner or a new strategy.

7.2.2 Strategies in MOPS

Experiment strategies in MOPS describe the way in which we use the metric sensitive planner to obtain a pareto-frontier. The basic strategies are using weighted sum of different objective functions to direct a planner into various areas of the search space and re-running a stochastic planner with all objective functions to obtain different results each run. All results in both cases are merged, dominated plans are removed and the resulting set is the set of non-dominated plans, with regards to selected objectives.

7.2.3 Compiling MOPS

The MOPS source code comes with a CMake file. So you only need to do:

```
cmake .
make
```

and the magic should happen automatically. The libraries required to compile it (listed in CMake file) are boost_regex boost_filesystem boost_system pthread rt dl. Running MOPS Along with MOPS we publish sample multi-objective domains and problem files which can be downloaded from here and are described below. An example command to run MOPS can be: *./mops -pf input/pfile.pddl -d input/domain.pddl -o output_plans -pif planIndex.txt -strategy mr -pp* LPG*-td*
Where the options are:

- -pf problem file with multiple objectives.
- -d name of a file containing a PDDL domain
- -o name of a folder where non dominated plans are stored

- -pif name of a file where an information about the plans stored in the output folder is saved
- -strategy name of the strategy to be used. In a standard version there are two strategies available: mr (multiple re-run) and ws (weighted sum of objectives).
- -n When mr strategy is used this option allows to specify how many times a planner should be re-run to generate pareto-frontier.
- -pp name of the planner registered in a planner factory. Standard version of MOPS comes with LPG-td planner.

7.2.4 Adding a new planner

MOPS loads planners as dynamic libraries. The libraries with planners should be in planners/ folder and have .so extension.

When MOPS is started is searches the planners/ folder for all .so files and expects them to load themselves into the factory like pattern, using std::map factory. The factory map is defined in iPlanner.h file provided with MOPS. Each planner should extend I_Planner interface and in the file which creates the planner, it should contain the following to register the new planner (Example for LPG-td planner, where C_LPGTD is the classname.)

```
extern "C" {
shape *maker(){
   return new C\_LPGTD;
}
class proxy {
public:
   proxy(){
```

```
      // register the maker with the factory used in MOPS
      factory["lpg-td"] = maker;
   }
};
// our, local, instance of the proxy
proxy p;
}
```

If you are using c_lpg_td.h for a header file and c_lpg_td.cpp for the class file, please compile it as follows:

g++ -c -o c_lpg_td.o c_lpg_td.cpp -fPIC

g++ -shared -Wl,-soname,lib_lpg_td.so -o lib_lpg_td.so c_lpg_td.o

Then, place the lib_lpg_td.so file in planners/ folder, and MOPS will automatically detect it. If you want to use your planner, when running MOPS add "-pp lpg-td" to it's parameters. Then from dynamically loaded map, it selects the planner which is stored with "lpg-td" key.

7.3 Appendix C. Metric sensitivity results

This section contains the following Tables: 7.1, 7.2, 7.3, 7.4, 7.5, 7.6, 7.7, 7.8, 7.9, 7.10, 7.11, 7.12, 7.13, 7.14, 7.15, 7.16.

The Results for the quality experiment for domain Bread. An average over 66 different weights is presented for each problem are presented in Table 7.1. The Results for the number of best plans generated for domain Bread. A total number of 66 different weights is presented for each problem are presented in Table 7.2. The Statistical results for 10 repetitions of the quality experiment for domain Bread are presented in Table 7.3. The Statistical results for 10 repetitions

of the quality experiment for domain Bread The table presents the total amount of best plans in each experiment. are presented in Table 7.4. The Results for the quality experiment for domain Production. An average over 66 different weights is presented for each problem are presented in Table 7.5. The Results for the number of best plans generated for domain Production. A total number of 66 different weights is presented for each problem are presented in Table 7.6. The Statistical results for 10 repetitions of the quality experiment for domain Production are presented in Table 7.7. The Statistical results for 10 repetitions of the quality experiment for domain Production The table presents the total amount of best plans in each experiment. are presented in Table 7.8. The Results for the quality experiment for domain Driverlog. An average over 66 different weights is presented for each problem are presented in Table 7.9. The Results for the number of best plans generated for domain Driverlog. A total number of 66 different weights is presented for each problem are presented in Table 7.10. The Statistical results for 10 repetitions of the quality experiment for domain Driverlog. are presented in Table 7.11. The Statistical results for 10 repetitions of the quality experiment for domain Driverlog. The table presents the total amount of best plans in each experiment are presented in Table 7.12. The Results for the quality experiment for domain **Driverlog Metric**. An average over 66 different weights is presented for each problem are presented in Table 7.13. The Results for the number of best plans generated for domain **Driverlog Metric**. A total number of 66 different weights is presented for each problem are presented in Table 7.14. The Statistical results for 10 repetitions of the quality experiment for domain Driverlog Metric are presented in Table 7.15. The Statistical results for 10 repetitions of the quality experiment for domain Driverlog Metric The table presents the total amount of best plans in each experiment are presented in Table 7.16.

7.4 Appendix D. Approximate pareto frontier generation results

This section contains the following Tables: 7.4, 7.17, 7.18, 7.20, 7.20, 7.21, 7.22, 7.23, 7.24.

TABLE 7.1: Results for the quality experiment for domain Bread. An average over 66 different weights is presented for each problem.

	LPG n1			LPG n2			LPG n3			MS-POPF2				
	St	Det	Com	St	Det	Com	St	Det	Com	Stoch	Det	StCod	StCoi	DeCo
1	9.18	11.09	9.7	4.59	5.07	2.62	1.45	1.67	0.74	12.56	12.56	13.78	1.62	13.78
2	11.41	11.66	10.96	4.86	3.43	3.69	1.9	1.43	1.58	15.55	15.19	16.26	1.56	16.26
3	3.23	3.92	4.55	1.2	1.73	1.68	0.46	0.82	0.42	4.58	4.58	4.58	1.02	4.58
4	3.21	3.86	4.22	1.13	1.71	1.59	0.16	0.8	0.59	4.47	4.47	4.47	1.01	4.47
5	7.61	8.44	7.9	2.63	3.33	2	1.72	2.17	0.56	8.36	8.21	9.17	0.76	8.62
6	7.59	8.61	7.46	4.72	3.71	3.22	1.68	2.22	1.31	6.99	8	8.78	0.76	8.35
7	5.05	4.48	5.02	1.6	1.48	1.08	0.66	0.51	0.21	2.99	2.88	4.77	1.39	4.39
8	8.02	6.75	9.49	2.05	1.05	3.7	0.7	0.3	0.99	9.51	9.28	10.12	0.85	10.08
9	8.35	7.73	10.18	4.3	4.53	1.99	1.62	2.18	1.37	6.65	6.65	9.14	0.65	9.14
10	9.71	10.65	8.03	3.93	4.41	4.35	1.4	2.89	0.78	8.77	8.76	9.86	0.89	9.86
11	11.59	8.11	13.27	4.09	2.36	4.82	1.43	1.32	1.01	8.29	7.67	10.44	0.95	9.99
12	11.72	11.59	12.12	4.15	3.98	2.75	1.97	1.25	1.74	9.83	7.73	11.22	1.14	11.23
13	12.71	11.2	14.13	5.87	4.26	5.89	3.79	2.74	1.81	9.74	8.79	9.63	0.39	9.76
14	13.25	14.73	14.76	5.26	4.46	5.16	2.06	2.7	0.72	10.84	8.84	10.93	1.09	10.73
15	21.72	21.71	20.49	5.5	3.07	6.71	3.53	1.18	2.93	6.99	3.38	15.73	0.91	14.49
16	10.52	10.75	11.69	3.96	4.97	3.93	2.14	1.98	0.86	8.91	8.67	13.71	0.54	13.6
17	9.13	6.88	7.99	4.09	2.68	3.33	2.76	1.87	3.68	8.41	8.52	10.38	0.3	10.18
18	22.57	22.22	24.81	6.3	5.75	6.94	2	3.24	3.39	10.88	6.98	20.36	1.37	20.36
19	14.49	16.33	14.35	3.1	4.76	4.11	2.38	3.02	2.2	7.77	7.68	14.53	0.58	13.87
20	28.5	28.6	28.64	8.58	13.61	7.93	3.86	6.97	8.71	10.36	7.48	23.77	1.35	23
Avg	11.478	11.4655	11.988	4.0955	4.0175	3.8745	1.8835	2.063	1.78	8.6225	7.816	11.5815	0.9565	11.337

TABLE 7.2: Results for the number of best plans generated for domain Bread. A total number of 66 different weights is presented for each problem.

	LPG n1			LPG n2			LPG n3			MS-POPF2				
	St	Det	Com	St	Det	Com	St	Det	Com	Stoch	Det	StCod	StCoi	DeCo
1	1	0	2	16	9	26	44	27	37	8	8	2	16	2
2	0	0	1	18	13	11	33	38	26	4	8	2	16	2
3	19	0	5	47	35	31	59	48	51	19	19	19	38	19
4	9	0	8	40	33	34	59	46	48	19	19	19	37	19
5	1	1	0	12	11	9	30	29	27	0	0	1	23	1
6	0	1	0	9	10	7	36	27	26	0	0	0	22	0
7	4	0	3	19	22	19	44	44	37	6	12	2	16	4
8	0	0	1	12	18	5	36	47	15	1	0	4	20	4
9	0	0	2	17	11	6	37	28	23	12	12	1	30	1
10	1	0	0	11	9	5	39	25	23	1	1	1	22	1
11	1	0	1	13	9	10	32	33	26	1	1	0	21	0
12	0	0	0	12	10	9	30	28	25	3	5	0	21	0
13	1	0	0	13	12	7	27	32	27	1	3	0	27	0
14	0	0	0	11	4	5	24	29	22	1	2	1	25	1
15	0	0	0	12	8	6	26	21	14	2	20	7	32	7
16	0	0	1	4	1	1	24	22	16	10	12	1	36	0
17	1	0	1	3	3	1	11	15	4	5	9	4	34	4
18	0	0	0	8	17	8	26	31	19	2	19	3	33	3
19	1	0	1	9	2	4	21	10	12	13	12	2	29	2
20	1	0	0	7	2	5	17	16	12	2	16	2	34	7
Sum	40	2	26	293	239	209	655	596	490	110	178	71	532	77

TABLE 7.3: Statistical results for 10 repetitions of the quality experiment for domain Bread.

	LPG n1			LPG n2			LPG n3			MS-POPF2				
	St	Det	Com	St	Det	Com	St	Det	Com	Stoch	Det	StCod	StCoi	DeCo
1	11.48	11.46	11.99	4.1	4.02	3.87	1.88	2.06	1.78	8.62	7.82	11.58	0.96	11.34
2	11.36	11.16	11.62	3.93	3.76	3.51	1.71	1.89	1.53	8.49	7.57	11.4	0.9	11.19
3	11.27	11.17	11.51	3.92	3.84	3.47	1.75	1.9	1.56	8.22	7.59	11.22	0.9	11
4	10.98	10.6	10.75	3.83	3.62	3.18	1.72	1.79	1.64	7.99	7.26	10.78	0.81	10.57
5	11.1	11.12	11.31	3.81	3.84	3.3	1.73	1.86	1.79	8.25	7.59	11.22	0.9	11
6	10.95	11.11	11.31	3.75	3.84	3.51	1.71	1.88	1.59	8.56	7.81	11.28	0.93	11.05
7	11.24	11.19	12.11	4.06	3.92	3.67	1.98	1.95	1.34	8.25	7.73	11.31	0.92	11.09
8	11.93	11.74	12.48	4.06	4.19	3.66	1.86	2.14	1.71	8.93	8.21	11.9	1.02	11.66
9	11.07	11.18	11.13	3.77	3.93	3.35	1.96	2.01	1.6	8.46	7.69	11.33	0.92	11.12
10	12.03	11.57	11.7	4.16	4.02	3.69	1.9	2.07	1.42	8.64	7.78	11.73	0.93	11.48
Avg	11.34	11.23	11.59	3.94	3.9	3.52	1.82	1.96	1.6	8.44	7.7	11.37	0.92	11.15

TABLE 7.4: Statistical results for 10 repetitions of the quality experiment for domain Bread. The table presents the total amount of best plans in each experiment.

	LPG n1			LPG n2			LPG n3			MS-POPF2				
	St	Det	Com	St	Det	Com	St	Det	Com	Stoch	Det	StCod	StCoi	DeCo
1	40	2	26	293	239	209	655	596	490	110	178	71	532	77
2	33	2	20	297	243	248	629	595	470	104	184	68	535	74
3	32	2	24	286	237	241	648	596	509	115	175	72	529	76
4	33	2	21	301	240	251	638	590	507	115	168	72	530	77
5	39	2	19	304	238	219	659	589	483	98	179	69	523	75
6	40	2	20	309	240	237	640	590	499	110	179	71	531	74
7	27	2	21	299	240	224	640	601	504	120	169	69	529	74
8	31	2	14	303	240	215	657	590	517	109	173	70	518	74
9	27	2	19	293	239	250	625	598	514	123	180	72	520	77
10	36	2	17	296	238	223	618	597	519	112	181	70	536	76
sum	338	20	201	2981	2394	2317	6409	5942	5012	1116	1766	704	5283	754

TABLE 7.5: Results for the quality experiment for domain Production. An average over 66 different weights is presented for each problem.

	LPG n1			LPG n2			LPG n3			MS-POPF2				
	St	Det	Com	St	Det	Com	St	Det	Com	Stoch	Det	StCod	StCoi	DeCo
1	11.96	4.38	11.52	5.61	3.59	6.4	3.24	10.57	5.72	15.13	17.96	8.39	0.03	9.96
2	14.53	10.43	13.18	10.54	8.63	10.84	9.63	13.05	9.85	15.36	16.33	8.13	0.01	8.09
3	14.75	5.92	11.48	7.05	5.13	10.68	4.98	15.74	5.89	19.73	21.59	8.04	0.02	8.5
4	22.18	15.03	18.62	11.84	11.44	15.67	10.14	22.61	16.61	24.46	27.33	10.16	0.01	7.56
5	14.1	9.43	15.46	11.53	9.05	13.02	12.66	13.71	13.07	23.32	14.07	7.24	0.01	7.17
6	2.8	0.9	2.63	0.27	0.09	0.67	0.03	1.29	0.09	2.13	2.13	1.76	0.3	1.85
7	19.52	16.63	19.55	17.65	15.55	15.11	15.3	19.15	15.72	19.24	19.27	8.75	0.02	8.75
8	2.82	0.06	2.53	0.04	0.02	0.17	0.01	2.28	0.05	0.93	0.65	0.4	0.17	0.38
9	20.19	14.92	21.7	14.71	14.1	19.98	13.97	22.39	17.13	22.11	22.09	10.28	0.03	10.43
10	30.57	4.95	27.01	4.56	0.85	5.85	1.17	25.33	2.79	36.9	37.4	10.87	1.11	11.04
11	30.01	3.49	25.78	2.71	1.18	9.17	1.19	24.53	2.07	30.92	34.73	11.38	1.68	11.54
12	13.97	9.14	16.58	8.42	6.71	12.08	8.19	13.43	10.45	17.95	18.29	10.26	0.02	9.74
13	6.39	3.2	6.35	3.67	1.23	3.21	2.47	6.53	2.08	6.53	6.53	6.53	0.17	6.53
14	2.44	0.24	3.11	0.32	0.1	0.53	0.1	2.17	0.21	2.92	2.92	1.99	0.37	1.99
15	12.3	8.46	10.84	8.36	6.21	7.8	5.97	11.94	7.03	14.04	11.94	8.5	0.04	8.74
16	12.42	8.46	13.86	8.65	6.22	10.85	7.8	11.95	9.88	12.3	11.95	8.51	0.05	8.74
17	14.64	5.51	14.78	5.65	4.89	9.39	5.11	11.82	7.4	15.86	15	8.82	0.01	7.52
18	10.42	9.76	10.25	8.29	6.97	8.48	6.52	10.07	6.49	10.65	10.12	7.59	0.01	7.6
19	3.1	0.23	3.31	0.19	0.11	0.35	0.07	3.3	0.13	1.7	1.11	1.84	0.15	0.91
20	21.38	17.6	21.52	15.57	15.42	18.31	13.25	22.2	13.91	28.71	32.11	5.33	0.03	5.67
Avg	14.02	7.43	13.50	7.28	5.87	8.92	6.09	13.20	7.33	16.04	16.18	7.24	0.21	7.14

TABLE 7.6: Results for the number of best plans generated for domain Production. A total number of 66 different weights is presented for each problem.

	LPG n1			LPG n2			LPG n3			MS-POPF2				
	St	Det	Com	St	Det	Com	St	Det	Com	Stoch	Det	StCod	StCoi	DeCo
1	0	2	0	5	10	3	8	0	5	0	0	0	54	0
2	3	3	1	7	7	2	8	2	6	0	0	1	57	1
3	0	1	0	3	6	0	8	0	4	1	1	0	55	0
4	0	2	0	4	2	1	4	0	2	0	2	0	60	0
5	0	6	0	5	8	2	6	3	3	0	0	1	58	1
6	6	31	2	29	56	19	51	6	41	6	6	6	31	6
7	4	7	0	6	10	3	11	1	7	0	0	0	56	0
8	6	50	5	52	57	43	59	5	50	32	35	23	37	23
9	3	4	2	3	5	3	8	0	3	0	0	0	57	0
10	1	3	0	7	20	1	27	0	8	0	1	0	24	0
11	0	4	0	5	21	2	25	0	8	1	1	0	23	0
12	0	2	0	4	7	3	5	0	2	0	0	0	56	0
13	11	20	11	19	47	22	30	11	39	11	11	11	56	11
14	1	28	2	27	46	24	46	1	37	1	1	1	38	1
15	0	2	0	4	5	1	10	0	2	1	0	0	55	0
16	0	1	0	4	3	2	7	0	6	0	0	0	56	0
17	0	2	0	2	9	0	6	0	4	1	0	0	54	0
18	1	5	2	4	15	3	13	2	5	0	2	4	43	3
19	2	19	1	23	27	18	33	1	28	6	11	1	36	11
20	0	0	0	3	4	0	4	0	2	2	3	0	56	0
Sum	38	192	26	216	365	152	369	32	262	62	74	48	962	57

TABLE 7.7: Statistical results for 10 repetitions of the quality experiment for domain Production.

	LPG n1			LPG n2			LPG n3			MS-POPF2				
	St	Det	Com	St	Det	Com	St	Det	Com	Stoch	Det	StCod	StCoi	DeCo
1	14.02	7.44	13.5	7.28	5.87	8.93	6.09	13.2	7.33	16.04	16.18	7.24	0.21	7.14
2	13.5	7.52	13.86	7.47	5.86	8.56	6.23	13.15	7.59	16.7	16.1	7.2	0.24	7.11
3	13.87	7.45	12.97	7.28	5.88	9.14	5.67	13.09	7.12	15.25	15.91	7.25	0.25	7.15
4	14	7.42	13.08	7.29	5.86	8.67	6.1	12.87	7.29	15.09	15.57	7.15	0.24	7.05
5	13.7	7.48	13.19	7.65	5.9	9.08	5.92	13.15	7.55	15.67	15.94	7.31	0.28	7.2
6	13.77	7.56	13.78	7.71	5.86	8.31	5.84	13.17	7.16	15.66	16.06	7.25	0.25	7.15
7	12.43	7.42	13.26	7.53	5.86	9.45	6.31	13.11	7.32	15.43	15.95	7.23	0.25	7.13
8	13.19	7.48	13.36	7.51	5.86	8.46	6.12	13.04	6.95	16.06	15.92	7.16	0.19	7.06
9	14.22	7.52	13.48	7.81	5.88	9.05	5.72	13.33	7.02	15.81	16.3	7.33	0.27	7.22
10	13.76	7.56	13.06	7.27	5.88	8.96	5.66	13.13	6.98	15.61	15.91	7.3	0.29	7.19
Avg	13.65	7.49	13.35	7.48	5.87	8.86	5.97	13.12	7.23	15.73	15.98	7.24	0.25	7.14

TABLE 7.8: Statistical results for 10 repetitions of the quality experiment for domain Production. The table presents the total amount of best plans in each experiment.

	LPG n1			LPG n2			LPG n3			MS-POPF2				
	St	Det	Com	St	Det	Com	St	Det	Com	Stoch	Det	StCod	StCoi	DeCo
1	38	192	26	216	365	152	369	32	262	62	74	48	962	57
2	31	191	26	212	361	149	387	34	266	58	76	51	967	60
3	32	195	32	229	365	151	373	36	279	52	73	50	961	59
4	37	191	32	209	365	163	372	33	268	66	74	47	955	56
5	28	195	27	206	361	156	387	34	275	49	75	48	964	57
6	34	191	25	215	365	169	378	35	260	60	72	48	974	57
7	30	194	26	211	361	151	376	34	261	51	74	47	976	56
8	36	196	31	212	362	153	379	34	281	45	73	50	970	59
9	34	191	27	201	368	157	379	33	279	58	74	47	961	56
10	35	191	28	228	358	150	376	34	257	59	73	49	963	58
Sum	335	1927	280	2139	3631	1551	3776	339	2688	560	738	485	9653	575

TABLE 7.9: Results for the quality experiment for domain Driverlog. An average over 66 different weights is presented for each problem.

	LPG n1			LPG n2			LPG n3			MS-POPF2				
	St	Det	Com	St	Det	Com	St	Det	Com	Stoch	Det	StCod	StCoi	DeCo
1	3	0.04	567	0.54	0.05	0.94	0.25	246	0.15	181	112	0.67	0.48	0.48
2	295	12	309	678	4	9	226	588	3	1032	773	3	0.56	0.56
3	1263	410	1907	648	168	389	204	659	107	1976	2331	0.2	0.09	0.09
4	1471	1607	1875	1056	693	1198	599	2524	287	2488	3093	1	0.72	0.72
5	356	169	303	167	33	416	18	226	323	1618	1096	0.79	0.28	0.28
6	145	0.42	290	1	0.16	1	0.09	140	0.2	153	153	1	1	1
7	602	0.25	993	609	0.06	11	281	0.44	1	519	505	1	0.62	0.62
8	4561	2311	5299	902	833	3341	1112	4652	1809	5143	2368	1	0.85	0.85
9	841	472	443	466	113	8	146	1456	2	379	916	0.87	0.97	0.97
10	3712	1	2294	1079	1	1561	521	2	554	3911	5835	0.53	0.42	0.42
11	5005	1	4123	3017	1	2934	1382	275	1540	5487	5613	0.62	0.46	0.46
12	1759	1	798	337	0.47	9	174	2	135	1676	1156	1	0.47	0.47
13	7639	935	7251	3070	57	4939	2726	1311	2709	8482	8525	1694	0.52	0.52
14	4904	2010	7704	4127	1025	3230	1689	4021	3470	7486	7635	1974	0.52	0.52
15	1785	540	1010	1006	1	521	30	1209	972	1980	0	52	0.58	0.58
Avg	2289	564	2344	1144	195	1238	607	1154	794	2834	2865	248	0.58	0.58

TABLE 7.10: Results for the number of best plans generated for domain Driverlog. A total number of 66 different weights is presented for each problem.

	LPG n1			LPG n2			LPG n3			MS-POPF2				
	St	Det	Com	St	Det	Com	St	Det	Com	Stoch	Det	StCod	StCoi	DeCo
1	9	55	12	37	45	35	41	5	49	11	37	21	21	21
2	5	5	7	4	22	16	21	2	35	1	0	7	16	16
3	2	13	1	8	30	7	24	1	22	3	0	36	40	40
4	4	26	3	16	30	13	21	2	20	0	0	21	35	35
5	2	4	3	11	20	12	23	2	22	1	2	25	36	36
6	9	35	8	30	52	30	48	11	45	12	12	16	22	22
7	2	12	1	10	42	5	21	2	23	1	1	2	21	21
8	7	12	6	19	24	14	23	11	14	8	32	21	25	25
9	3	1	3	10	10	6	17	0	21	2	0	19	10	10
10	6	23	6	13	23	17	20	21	20	10	0	30	29	29
11	7	21	8	13	25	14	26	2	21	1	1	25	22	22
12	2	17	2	10	27	12	13	6	11	1	0	5	12	12
13	2	6	1	7	16	4	16	1	18	0	0	3	23	23
14	3	8	3	12	24	10	19	1	14	0	0	5	23	23
15	1	3	3	9	13	8	3	1	11	0	0	6	32	32
Sum	64	241	67	209	403	203	336	68	346	51	85	242	367	367

TABLE 7.11: Statistical results for 10 repetitions of the quality experiment for domain Driverlog.

	LPG n1			LPG n2			LPG n3			MS-POPF2				
	St	Det	Com	St	Det	Com	St	Det	Com	Stoch	Det	StCod	StCoi	DeCo
1	2289	564	2344	1144	195	1238	607	1154	794	2834	2865	248	0.58	0.58
2	2309	564	2280	1474	195	1039	626	1154	607	2865	2865	248	0.59	0.59
3	2639	565	2563	1246	195	1300	552	1154	599	2759	2866	248	0.58	0.58
4	2166	565	2594	1299	195	1233	634	1154	636	2791	2866	248	0.57	0.57
5	1590	415	1343	949	154	702	292	898	239	1982	1997	1	0.62	0.61
6	2359	565	2159	1390	195	1251	770	1154	629	2656	2865	248	0.59	0.59
7	2312	565	2414	1390	195	1392	748	1154	725	2657	2865	248	0.59	0.59
8	2226	565	2324	1177	195	1126	589	1154	604	3070	2866	248	0.58	0.58
9	2574	565	2637	1249	195	1392	631	1154	628	2674	2866	248	0.59	0.59
10	2347	565	2577	1379	195	1583	657	1154	771	3006	2866	248	0.59	0.59
Avg	2281	550	2323	1270	191	1226	611	1128	623	2729	2779	223	0.59	0.59

TABLE 7.12: Statistical results for 10 repetitions of the quality experiment for domain Driverlog. The table presents the total amount of best plans in each experiment.

	LPG n1			LPG n2			LPG n3			MS-POPF2				
	St	Det	Com	St	Det	Com	St	Det	Com	Stoch	Det	StCod	StCoi	DeCo
1	64	241	67	209	403	203	336	68	346	51	85	242	367	367
2	62	241	76	201	400	226	335	68	400	50	84	231	361	361
3	57	249	72	210	405	213	350	70	391	50	87	233	366	366
4	75	243	72	205	396	235	371	68	363	61	83	235	367	367
5	60	228	66	182	349	198	323	67	333	47	86	218	274	285
6	77	245	80	227	400	221	344	71	399	66	88	236	363	363
7	68	241	76	209	402	223	344	70	371	66	86	231	365	365
8	71	254	77	227	405	226	352	69	395	58	83	234	364	364
9	69	245	75	208	397	208	352	69	379	56	85	232	368	368
10	71	240	84	193	393	215	374	68	367	57	86	236	360	360
Avg	674	2427	745	2071	3950	2168	3481	688	3744	562	853	2328	3555	3566

TABLE 7.13: Results for the quality experiment for domain **Driverlog Metric**. An average over 66 different weights is presented for each problem.

	LPG n1			LPG n2			LPG n3			MS-POPF2				
	St	Det	Com	St	Det	Com	St	Det	Com	Stoch	Det	StCod	StCoi	DeCo
1	242	0.15	235	105	0.11	1	0.4	140	0.23	181	112	0	0	0
2	625	108	592	212	4	587	2	588	0.5	575	774	4	0.78	0.78
3	913	410	1149	649	103	493	273	691	132	1944	2332	1	0.03	0.03
4	1514	1495	1712	579	364	730	359	2590	285	3042	3095	1	0.27	0.27
5	395	49	363	114	30	109	335	225	15	1569	1096	1	0.24	0.24
6	151	0.57	294	0.5	0.22	1	0.22	144	0.2	160	158	4	2	2
7	5	0.9	558	82	0.42	4	5	1	2	493	508	2	0.81	0.81
8	4723	2366	4113	1353	463	2911	546	4657	1751	4086	2372	2	0.01	0.01
9	10	500	8	397	149	4	2	1456	503	593	917	0.91	0.78	0.78
10	3259	1	2432	1644	1	1011	281	2	547	4224	5835	0.53	0.43	0.43
11	4224	937	6582	2816	124	3487	3165	2966	2099	6059	7121	0.81	0.44	0.44
12	862	13	1986	494	7	8	1	31	3	201	1156	0.52	0.44	0.44
13	6486	935	6753	5217	57	4896	2965	1311	1960	7196	8525	1694	0.47	0.47
Avg	1801	524	2060	1051	100	1096	610	1139	561	2333	2616	131	0.56	0.56

TABLE 7.14: Results for the number of best plans generated for domain **Driverlog Metric**. A total number of 66 different weights is presented for each problem.

	LPG n1			LPG n2			LPG n3			MS-POPF2				
	St	Det	Com	St	Det	Com	St	Det	Com	Stoch	Det	StCod	StCoi	DeCo
1	7	36	9	17	40	23	30	1	34	11	11	66	66	66
2	2	3	7	5	25	15	18	1	32	0	0	16	16	16
3	1	7	1	1	21	4	18	1	18	1	0	24	55	55
4	2	12	2	13	23	14	18	1	19	0	0	23	33	33
5	4	19	3	8	27	11	22	2	24	2	3	23	32	32
6	1	30	1	33	51	25	43	1	51	1	1	2	3	3
7	2	14	1	8	35	8	17	2	19	1	1	5	26	26
8	7	11	10	14	15	10	16	11	14	5	11	24	63	63
9	4	1	5	7	8	12	12	0	24	1	0	10	10	10
10	7	22	9	11	22	17	23	21	25	8	0	30	29	29
11	9	3	8	11	13	23	26	0	20	0	0	21	27	27
12	2	3	2	11	9	15	18	2	22	2	0	7	5	5
13	4	7	2	5	19	8	12	1	19	0	0	3	24	24
Sum	52	168	60	144	308	185	273	44	321	32	27	254	389	389

TABLE 7.15: Statistical results for 10 repetitions of the quality experiment for domain **Driverlog Metric**.

	LPG n1			LPG n2			LPG n3			MS-POPF2				
	St	Det	Com	St	Det	Com	St	Det	Com	Stoch	Det	StCod	StCoi	DeCo
1	1801	524	2060	1051	100	1096	610	1139	561	2333	2616	131	0.56	0.56
2	2240	525	1735	1169	100	810	663	1139	461	2432	2616	131	0.56	0.56
3	2454	524	1827	909	100	965	546	1139	624	2349	2616	131	0.55	0.55
4	2218	524	1965	1003	100	1030	732	1139	541	2557	2616	131	0.58	0.58
5	2256	524	2085	1478	100	1128	426	1139	579	2450	2616	131	0.57	0.57
6	2645	524	2142	1103	100	1218	507	1139	419	2372	2616	131	0.57	0.57
7	1833	525	2238	934	100	1040	425	1140	305	2431	2616	131	0.57	0.57
8	2048	524	2651	1015	100	952	512	1139	477	2541	2616	131	0.57	0.57
9	2451	524	2242	1123	100	858	412	1139	501	2637	2615	131	0.55	0.55
10	2466	524	2035	1182	100	1068	467	1139	780	2464	2616	131	0.57	0.57
Avg	2241	524	2098	1097	100	1017	530	1139	525	2457	2616	131	0.57	0.57

TABLE 7.16: Statistical results for 10 repetitions of the quality experiment for domain **Driverlog Metric**. The table presents the total amount of best plans in each experiment.

	LPG n1			LPG n2			LPG n3			MS-POPF2				
	St	Det	Com	St	Det	Com	St	Det	Com	Stoch	Det	StCod	StCoi	DeCo
1	52	168	60	144	308	185	273	44	321	32	27	254	389	389
2	53	173	59	154	303	181	313	45	301	32	26	253	388	388
3	49	171	60	161	303	160	303	44	306	32	26	259	385	385
4	45	159	46	166	300	155	296	44	322	32	25	259	392	392
5	44	171	54	161	307	174	302	44	299	30	25	251	384	384
6	38	173	47	172	316	170	292	45	310	36	26	252	393	393
7	57	166	49	158	305	179	320	44	304	37	26	254	384	384
8	52	170	53	173	300	181	287	44	301	37	25	256	388	388
9	45	168	46	156	306	192	291	45	291	35	25	258	387	387
10	45	168	61	171	306	173	332	44	284	28	25	250	386	386
Avg	480	1687	535	1616	3054	1750	3009	443	3039	331	256	2546	3876	3876

Table 7.4 Represents the Results for the quality, expressed as NDHVI, experiment for domain Bread. An average over 66 different weights is presented for each problem. Table 7.17 Represents the Statistical results for 10 repetitions of the quality, expressed as NDHVI, experiment for domain Bread. Table 7.18 Represents the Results for the quality experiment for domain Production. An average over 66 different weights is presented for each problem. Table 7.20 Represents the Results for the number of best plans generated for domain Production. A total number of 66 different weights is presented for each problem. Table 7.20 Represents the Results for the quality experiment for domain Driverlog. An average over 66 different weights is presented for each problem. Table 7.21 Represents the Statistical results for 10 repetitions of the quality experiment for domain Driverlog. Table 7.22 Represents the Results for the quality experiment for domain **Driverlog Metric**. An average over 66 different weights is presented for each problem. Table 7.23 Represents the Statistical results for 10 repetitions of the quality experiment for domain **Driverlog Metric**. Table 7.24

TABLE 7.17: Results for the quality, expressed as NDHVI, experiment for domain Bread. An average over 66 different weights is presented for each problem.

	LPG n1			LPG n2			LPG n3			MS-POPF2				
	St	Det	Com	St	Det	Com	St	Det	Com	Stoch	Det	StCod	StCoi	DeCo
1	0.5	2.16	0.43	0.02	0.44	0	0	0.04	0.02	2.31	2.31	2.35	0.35	2.35
2	0.61	1.56	0.41	0.22	0.17	0.04	0	0.07	0.04	1.88	1.95	1.98	0.44	1.98
3	0.41	1.19	0	0	0.41	0	0	0.04	0	1.22	1.22	1.22	0.41	1.22
4	0.41	1.19	0.07	0	0.41	0	0	0.04	0	1.22	1.22	1.22	0.41	1.22
5	0.81	1.66	0.65	0.26	0.18	0.16	0.12	0.03	0.03	1.19	1.52	1.51	0.23	1.56
6	1.22	1.55	0.56	0.37	0.19	0.13	0.02	0.04	0.09	1.15	1.45	1.47	0.23	1.5
7	0.26	0.57	0.11	0.07	0.08	0.06	0.02	0.06	0.02	0.19	0.78	0.37	0.22	0.53
8	0.49	0.82	0.24	0.2	0.16	0.19	0.05	0.11	0.06	1.13	1.15	1.16	0.33	1.16
9	0.85	2.27	0.52	0.09	0.19	0.17	0.04	0.04	0.04	2.81	2.81	2.77	0.38	2.77
10	0.49	2.38	0.66	0.18	0.65	0.2	0.03	0.14	0.13	2.89	2.97	2.2	0.38	2.2
11	0.5	1.09	0.41	0.13	0.14	0.11	0.09	0.08	0.05	1.05	1.31	1.09	0.38	1.13
12	0.6	1.19	0.36	0.15	0.19	0.13	0.08	0.11	0.05	1	1.27	1.04	0.37	1.13
13	0.6	1.32	0.85	0.27	0.2	0.22	0.09	0.1	0.13	1.33	1.45	1.19	0.47	1.19
14	0.71	1.34	0.57	0.21	0.2	0.13	0.06	0.16	0.12	1.07	1.29	1.1	0.41	1.1
15	0.54	1.17	0.41	0.12	0.16	0.16	0.05	0.27	0.08	0.51	1.11	0.96	0.48	0.96
16	0.33	1.04	0.46	0.15	0.43	0.17	0.05	0.12	0.08	1.3	1.31	1.19	0.47	1.19
17	0.4	1	0.48	0.29	0.17	0.21	0.13	0.14	0.13	1.4	1.4	1.24	0.44	1.24
18	0.6	0.95	0.24	0.06	0.06	0.06	0.03	0.04	0.02	0.58	0.85	0.85	0.32	0.85
19	0.72	1.39	0.55	0.14	0.21	0.16	0.08	0.15	0.1	0.95	1.04	1.2	0.32	1.22
20	0.79	1.08	0.56	0.15	0.21	0.11	0.06	0.14	0.09	0.43	0.89	0.97	0.48	0.98
Avg	0.592	1.346	0.427	0.154	0.2425	0.1205	0.05	0.096	0.064	1.2805	1.465	1.354	0.376	1.374

TABLE 7.18: Statistical results for 10 repetitions of the quality, expressed as NDHVI, experiment for domain Bread.

	LPG n1			LPG n2			LPG n3			MS-POPF2				
	St	Det	Com	St	Det	Com	St	Det	Com	Stoch	Det	StCod	StCoi	DeCo
1	0.59	1.35	0.43	0.15	0.24	0.12	0.05	0.09	0.06	1.28	1.46	1.35	0.38	1.37
2	0.56	1.27	0.45	0.14	0.24	0.12	0.05	0.1	0.06	1.21	1.36	1.25	0.37	1.27
3	0.57	1.28	0.45	0.12	0.23	0.12	0.05	0.09	0.06	1.2	1.37	1.28	0.37	1.3
4	0.58	1.3	0.46	0.11	0.24	0.11	0.04	0.09	0.07	1.24	1.4	1.29	0.37	1.31
5	0.58	1.32	0.47	0.14	0.23	0.11	0.04	0.1	0.06	1.24	1.41	1.33	0.38	1.35
6	0.57	1.34	0.47	0.12	0.24	0.13	0.05	0.09	0.06	1.31	1.47	1.34	0.37	1.36
7	0.58	1.3	0.47	0.12	0.24	0.11	0.04	0.1	0.07	1.24	1.41	1.29	0.37	1.32
8	0.57	1.33	0.47	0.12	0.24	0.13	0.04	0.09	0.06	1.28	1.46	1.34	0.37	1.36
9	0.58	1.31	0.46	0.12	0.25	0.11	0.05	0.1	0.06	1.22	1.39	1.29	0.37	1.31
10	0.53	1.3	0.49	0.11	0.24	0.13	0.05	0.1	0.05	1.23	1.4	1.29	0.37	1.31
Avg	0.57	1.31	0.46	0.13	0.24	0.12	0.05	0.1	0.06	1.25	1.41	1.31	0.37	1.33

TABLE 7.19: Results for the quality experiment for domain Production. An average over 66 different weights is presented for each problem.

	LPG n1			LPG n2			LPG n3			MS-POPF2				
	St	Det	Com	St	Det	Com	St	Det	Com	Stoch	Det	StCod	StCoi	DeCo
1	0.49	0.28	0.51	0.23	0.24	0.32	0.12	1.21	0.19	1.12	1.54	0.84	0.23	0.8
2	0.56	0.38	0.65	0.46	0.26	0.38	0.2	0.86	0.26	0.73	0.94	0.66	0.18	0.78
3	0.44	0.32	0.49	0.23	0.32	0.31	0.23	0.7	0.19	1.05	1.28	0.53	0.42	0.56
4	0.5	0.61	0.55	0.46	0.45	0.35	0.35	0.7	0.38	0.8	1.07	0.58	0.22	0.54
5	0.7	0.45	0.73	0.52	0.41	0.6	0.3	0.88	0.52	0.8	1.02	0.44	0.18	0.84
6	0.06	0	0.06	0	0.03	0	0	0.33	0	0.27	0.27	0.06	0.03	0.12
7	0.33	0.19	0.24	0.17	0.17	0.22	0.13	0.43	0.18	0.43	0.43	0.18	0.21	0.2
8	0.13	0.17	0.21	0.13	0.17	0.13	0.13	0.58	0.13	0.54	0.75	0.25	0	0.25
9	0.57	0.32	0.63	0.35	0.26	0.45	0.22	0.83	0.26	0.77	0.8	0.43	0.24	0.47
10	0.57	0.27	0.69	0.13	0.26	0.33	0.06	1.61	0.16	1.55	2.1	0.9	0.29	0.91
11	0.79	0.3	0.6	0.17	0.27	0.33	0.1	1.74	0.23	1.54	2.15	1.01	0.28	1.03
12	0.68	0.43	0.83	0.34	0.34	0.53	0.31	0.95	0.42	0.97	1.29	0.87	0.21	0.83
13	0	0	0	2	2	2	2	1	2	1	1	1	2	1
14	0.32	0.23	0.32	0.23	0.27	0.23	0.14	0.95	0.09	0.82	0.82	0.55	0.14	0.55
15	0.47	0.31	0.65	0.28	0.29	0.34	0.27	0.69	0.27	0.79	0.9	0.6	0.17	0.75
16	0.56	0.31	0.48	0.31	0.28	0.38	0.27	0.68	0.17	0.8	0.89	0.6	0.17	0.75
17	0.46	0.25	0.49	0.28	0.25	0.31	0.21	0.76	0.26	0.86	1.07	0.65	0.24	0.77
18	0.2	0.06	0.2	0.13	0.06	0.16	0.07	0.3	0.11	0.26	0.33	0.28	0.06	0.28
19	0.59	0.3	0.63	0.22	0.22	0.3	0.19	1.22	0.26	1.48	1.67	1.26	0.04	1.22
20	0.59	0.34	0.53	0.3	0.31	0.37	0.26	0.75	0.29	0.87	1.2	0.45	0.25	0.51
Avg	0.45	0.28	0.47	0.35	0.34	0.4	0.28	0.86	0.32	0.87	1.08	0.61	0.28	0.66

TABLE 7.20: Results for the number of best plans generated for domain Production. A total number of 66 different weights is presented for each problem.

	LPG n1			LPG n2			LPG n3			MS-POPF2				
	St	Det	Com	St	Det	Com	St	Det	Com	Stoch	Det	StCod	StCoi	DeCo
1	0.45	0.27	0.48	0.35	0.34	0.4	0.28	0.86	0.32	0.87	1.08	0.61	0.28	0.66
2	0.44	0.28	0.44	0.35	0.34	0.39	0.28	0.86	0.33	0.86	1.08	0.61	0.27	0.66
3	0.44	0.26	0.45	0.33	0.33	0.39	0.26	0.82	0.3	0.82	1.04	0.58	0.27	0.63
4	0.46	0.28	0.47	0.34	0.35	0.38	0.27	0.85	0.31	0.84	1.06	0.6	0.28	0.65
5	0.43	0.27	0.45	0.33	0.34	0.4	0.27	0.83	0.31	0.84	1.05	0.59	0.28	0.64
6	0.42	0.27	0.46	0.33	0.34	0.39	0.28	0.83	0.32	0.85	1.05	0.59	0.27	0.64
7	0.45	0.27	0.46	0.34	0.34	0.39	0.27	0.84	0.32	0.85	1.06	0.59	0.28	0.64
8	0.45	0.28	0.46	0.35	0.34	0.4	0.28	0.84	0.3	0.84	1.07	0.6	0.28	0.65
9	0.49	0.27	0.45	0.34	0.33	0.39	0.27	0.83	0.3	0.85	1.04	0.59	0.27	0.64
10	0.44	0.27	0.45	0.35	0.34	0.39	0.27	0.83	0.3	0.84	1.06	0.59	0.28	0.64
Avg	0.45	0.27	0.46	0.34	0.34	0.39	0.27	0.84	0.31	0.85	1.06	0.6	0.28	0.64

TABLE 7.21: Results for the quality experiment for domain Driverlog. An average over 66 different weights is presented for each problem.

	LPG n1			LPG n2			LPG n3			MS-POPF2				
	St	Det	Com	St	Det	Com	St	Det	Com	Stoch	Det	StCod	StCoi	DeCo
1	0	0	0	0	0	0	0	0	0	0.64	0.25	0.64	0	0
2	0.9	1.91	1.08	0.41	0.82	0.37	0.18	0.22	0.15	2.03	3.44	0.77	0.61	0.61
3	1.17	3.55	1.1	0.78	0.8	0.65	0.18	0.16	0.23	2.88	3.9	0.4	0.4	0.4
4	0.2	1.12	0.47	0.04	0.17	0.13	0.1	0.04	0.04	2.73	4.59	1.13	0.52	0.52
5	1.3	4.93	1.6	0.37	2.19	0.75	0.27	0.89	0.39	5.26	6.42	0.69	0.34	0.34
6	0.52	1.14	0.74	0	0.01	0	0	0.01	0	1.14	1.14	0.1	0.01	0.01
7	0.25	0.83	0.29	0.02	0.09	0.08	0.03	0.08	0.04	0.93	1.5	0.77	0.28	0.28
8	0.37	2.71	0.66	0.04	0.08	0.02	0.03	0.02	0.02	1.86	1.94	1.55	0.06	0.06
9	0.41	2.35	0.41	0.34	1.12	0.14	0.21	0.37	0.13	1.72	2.76	0.2	1.13	1.13
10	1.21	2	1.29	0.44	1.21	0.62	0.15	0.84	0.21	6.35	6.63	0.76	2.04	2.04
11	1.88	4.21	1.47	0.31	0.82	0.47	0.15	0.33	0.17	1.99	5.61	1.81	1.59	1.59
12	0.95	1.1	0.89	0.26	0.25	0.44	0.26	0.16	0.32	1.02	3.38	0.58	0.44	0.44
13	1.03	2.38	1.2	0.37	0.53	0.25	0.14	0.33	0.03	1.47	4.47	1.73	0.8	0.8
14	2.15	3.35	2.11	0.31	0.5	0.66	0.41	0.26	0.15	4.37	11.01	0.91	1.29	1.29
Avg	0.88	2.26	0.95	0.26	0.61	0.33	0.15	0.27	0.13	2.46	4.07	0.86	0.68	0.68

TABLE 7.22: Statistical results for 10 repetitions of the quality experiment for domain Driverlog.

	LPG n1			LPG n2			LPG n3			MS-POPF2				
	St	Det	Com	St	Det	Com	St	Det	Com	Stoch	Det	StCod	StCoi	DeCo
1	0.88	2.26	0.95	0.26	0.61	0.33	0.15	0.27	0.13	2.46	4.07	0.86	0.68	0.68
2	1.13	2.32	1.1	0.39	0.61	0.28	0.13	0.31	0.15	2.5	3.88	0.91	0.73	0.73
3	1.03	2.08	0.97	0.24	0.53	0.23	0.12	0.25	0.15	2.35	3.64	0.83	0.65	0.65
4	1.01	2.11	1.04	0.31	0.52	0.27	0.13	0.24	0.16	2.67	4.14	0.84	0.63	0.63
5	1	2.11	1.04	0.25	0.53	0.27	0.16	0.25	0.14	2.29	3.95	0.83	0.67	0.67
6	0.95	2.2	0.93	0.32	0.58	0.33	0.13	0.28	0.13	2.17	3.66	0.86	0.7	0.7
7	0.81	2.21	1.07	0.38	0.64	0.28	0.15	0.28	0.16	2.86	4.32	0.88	0.65	0.65
8	1.04	2.17	1.1	0.25	0.56	0.33	0.21	0.27	0.11	2.46	4.25	0.87	0.67	0.67
9	1.15	2.3	1.19	0.28	0.68	0.23	0.16	0.26	0.13	3.02	4.31	0.84	0.66	0.66
10	1.1	2.27	1.1	0.28	0.64	0.32	0.14	0.28	0.15	2.51	4.17	0.89	0.68	0.68
Avg	1.01	2.2	1.05	0.3	0.59	0.29	0.15	0.27	0.14	2.53	4.04	0.86	0.67	0.67

TABLE 7.23: Results for the quality experiment for domain **Driverlog Metric**. An average over 66 different weights is presented for each problem.

	LPG n1			LPG n2			LPG n3			MS-POPF2				
	St	Det	Com	St	Det	Com	St	Det	Com	Stoch	Det	StCod	StCoi	DeCo
1	0.06	0	0.18	0	0	0	0	0	0	0.34	0.18	0	0	0
2	2.79	3.88	2.68	1	1.37	1.09	0.69	0.57	0.4	3.21	6.11	0.72	0.71	0.71
3	5.28	7.72	3.26	2.12	2.17	2.42	0.76	0.67	0.55	6.7	8.71	4.18	0.18	0.18
4	2.81	2.94	2.93	0.76	1.12	1.19	0.46	0.44	0.42	13.35	21.54	1.49	0.09	0.09
5	3.59	5.4	4.72	0.78	0.71	0.83	0.32	0.46	0.53	6.23	7.02	4.05	0.99	0.99
6	1.45	2.23	1	0.18	1	0.27	0.03	0.03	0.05	2.23	2.23	0.01	0.01	0.01
7	0.75	1.63	0.92	0.18	0.31	0.13	0.16	0.15	0.02	1.5	1.95	0.57	0.46	0.46
8	28.61	63.72	37.57	10.19	14.94	10.72	4.56	8.39	3.41	54.76	50.05	11.81	0	0
9	1.21	3.51	1.16	0.6	1.46	0.41	0.62	0.67	0.12	2.2	5.31	0.55	0.83	0.83
10	1.69	1.95	1.68	0.44	1.19	0.5	0.24	0.87	0.31	4.69	9.56	0.81	2.23	2.23
11	2.47	4.02	1.92	0.5	0.47	0.36	0.29	0.22	0.21	2.77	6.71	2.49	1.49	1.49
12	0.66	1.07	1.02	0.34	0.25	0.32	0.16	0.19	0.07	1.09	3.67	0.55	0.78	0.78
13	1.41	2.7	1.41	0.59	0.64	0.22	0.17	0.39	0.2	2.52	6.53	1.98	0.88	0.88
Avg	4.06	7.75	4.65	1.36	1.97	1.42	0.65	1	0.48	7.81	9.97	2.25	0.67	0.67

TABLE 7.24: Statistical results for 10 repetitions of the quality experiment for domain **Driverlog Metric**.

	LPG n1			LPG n2			LPG n3			MS-POPF2				
	St	Det	Com	St	Det	Com	St	Det	Com	Stoch	Det	StCod	StCoi	DeCo
1	4.06	7.75	4.65	1.36	1.97	1.42	0.65	1.01	0.48	7.82	9.97	2.25	0.67	0.67
2	4.48	7.86	3.25	1.58	1.93	1.34	0.83	0.98	0.65	7.52	9.87	2.16	0.59	0.59
3	4.13	7.86	3.68	1.56	2	1.38	0.62	1.04	0.63	7.3	9.65	2.42	0.72	0.72
4	3.59	7.58	3.71	1.17	1.86	1.5	1.16	0.96	1.28	6.72	9.26	2.01	0.6	0.6
5	4.81	8.04	5.2	1.4	2.04	1.6	1.04	1.05	0.73	7.36	9.81	2.36	0.7	0.7
6	5.36	7.76	4.2	1.79	1.96	1.37	0.65	1.01	0.51	7.5	9.49	2.23	0.68	0.68
7	3.76	7.67	3.89	1.82	1.89	1.87	0.58	0.98	0.86	7.24	9.34	2.25	0.64	0.64
8	5.04	8	3.26	1.52	1.95	1.41	0.64	1	1.02	7.58	9.96	2.24	0.65	0.65
9	4.85	7.8	4.82	0.96	1.93	1.47	0.81	0.98	0.84	8.02	10.11	2.23	0.62	0.62
10	4.19	7.93	5.4	1.87	1.85	1.54	1.08	0.95	0.96	7.42	9.55	2.19	0.61	0.61
Avg	4.43	7.83	4.21	1.5	1.94	1.49	0.81	0.99	0.8	7.45	9.7	2.23	0.65	0.65

7.5 Appendix E. Published work

Exploring Metric Sensitivity of Planners for the Generation of Pareto Frontiers, 2012 [55]
Abstract This paper explores how current planners behave when exposed to multiple metrics, examining which of the planners are metric sensitive and which are not. For the metric insensitive planners we propose a new method of simulating metric sensitivity for the purpose of generation of diverse plans close to a pareto frontier. It is shown that metric sensitive planners are good candidates for generating sets of pareto optimal plans.

LPG **Based System for the Generation of Pareto Frontiers, 2012 [56]** **Abstract** In this paper we discuss the usefulness of multi-objective planning, its real life application, and example problems. A Planning system based on LPG planner is introduced. This system is capable of reasoning with multiple objectives and instead of a single solution it generates a set of high quality solutions. The quality of solution set is evaluated based on the users objectives. This is a more flexible way of specifying users preferences comparing to the single weighted metric function, used in planning nowadays. The new planning system is evaluated in terms of quality of the solution set and the results are presented.

Building a Metric Sensitive Planner, 2014 [57] **Abstract** Many current applications of planning depend on finding high quality solutions. In many cases finding the optimal solution is impractical, but a good estimation of it is required. The quality is evaluated in terms of user defined metric function. This function represent users preferences. We discuss the current state-of-the-art for finding good quality solutions and present a new way of generating high quality plans in response to the change of the metrics using a modified version of relaxed planning graph.

A Cost-Based Relaxed Planning Graph Heuristic for Enhanced Metric Sensitivity, 2014 [58]
Abstract Most applications of planning depend on finding high quality solutions. The quality

is evaluated in terms of user defined metric function. We discuss the current state-of-the-art for finding good quality solutions, and its limitations. We determine which planners are metric sensitive and are driven by cost. Following that we present a novel metric sensitivity heuristic using a modified version of the relaxed planning graph. The proposed heuristic helps in generating plans in response to the change of the metrics.

7.6 Appendix F. Domains

7.7 Driverlog

```
(define (domain driverlog)
  (:requirements :typing :fluents :equality :adl)
  (:types
    truck location locatable - object
    driver obj - locatable
    petroltruck electrictruck - truck
)

  (:predicates
(at ?obj - object ?loc - location)
(in ?obj1 - obj ?obj - truck)
(driving ?d - driver ?v - truck)
(link ?x ?y - location) (path ?x ?y - location)
(empty ?v - truck)
)
  (:functions  (time-to-walk ?l1 ?l2 - location)
```

```
(time-to-drive ?l1 ?l2 - location)
(fuel-used)
(fuel-per-minute ?t - petroltruck)
(electricity-used)
(electricity-per-minute ?t - electrictruck)
(driven)
(load ?t - truck)
(walked)
  )

(:action LOAD-TRUCK
  :parameters
   (?obj - obj
    ?truck - petroltruck
    ?loc - location)
  :precondition
   (and
      (at ?truck ?loc)
      (at ?obj ?loc)
   )
  :effect
   (and
      (not (at ?obj ?loc))
      (in ?obj ?truck)
      (increase (load ?truck) 1)
      (increase (fuel-per-minute ?truck) (+ (load ?truck) 1))
 )
```

```
)

(:action UNLOAD-TRUCK
  :parameters
   (?obj - obj
    ?truck - petroltruck
    ?loc - location)
  :precondition
   (and
      (at ?truck ?loc)
      (in ?obj ?truck)
   )
  :effect
   (and
      (not (in ?obj ?truck))
      (at ?obj ?loc)
      (decrease (load ?truck) 1)
      (decrease (fuel-per-minute ?truck) (load ?truck))
 )
)

(:action LOAD-ELECTRICTRUCK
  :parameters
   (?obj - obj
    ?truck - electrictruck
    ?loc - location)
  :precondition
```

```
   (and
      (at ?truck ?loc)
      (at ?obj ?loc)
   )
  :effect
   (and
      (not (at ?obj ?loc))
      (in ?obj ?truck)
      (increase (load ?truck) 1)
      (increase (electricity-per-minute ?truck) (load ?truck))
 )
)

(:action UNLOAD-ELECTRICTRUCK
  :parameters
   (?obj - obj
    ?truck - electrictruck
    ?loc - location)
  :precondition
   (and
      (at ?truck ?loc)
      (in ?obj ?truck)
   )
  :effect
   (and
      (not (in ?obj ?truck))
      (at ?obj ?loc)
```

```
      (decrease (load ?truck) 1)
      (decrease (electricity-per-minute ?truck) (load ?truck))
   )
)

(:action BOARD-TRUCK
  :parameters
   (?driver - driver
    ?truck - truck
    ?loc - location)
  :precondition
   (and
      (at ?truck ?loc)
      (at ?driver ?loc)
      (empty ?truck)
   )
  :effect
   (and
      (not (at ?driver ?loc))
      (driving ?driver ?truck)
      (not (empty ?truck))
   )
)

(:action DISEMBARK-TRUCK
  :parameters
   (?driver - driver
```

```
    ?truck - truck
    ?loc - location)
  :precondition
   (and
      (at ?truck ?loc)
      (driving ?driver ?truck)
   )
  :effect
   (and
      (not (driving ?driver ?truck))
      (at ?driver ?loc)
      (empty ?truck)
   )
)

(:action DRIVE-PETROLTRUCK
  :parameters
   (?truck - petroltruck
    ?loc-from - location
    ?loc-to - location
    ?driver - driver)
  :precondition
   (and
      (at ?truck ?loc-from)
      (driving ?driver ?truck)
      (link ?loc-from ?loc-to)
   )
```

```
  :effect
   (and
      (not (at ?truck ?loc-from))
      (at ?truck ?loc-to)
      (increase (fuel-used) (* (fuel-per-minute ?truck)
                               (time-to-drive ?loc-from ?loc-to)))
   )
)



(:action DRIVE-ELECTRICTRUCK
  :parameters
   (?truck - electrictruck
    ?loc-from - location
    ?loc-to - location
    ?driver - driver)
  :precondition
   (and
      (at ?truck ?loc-from)
      (driving ?driver ?truck)
      (link ?loc-from ?loc-to)
   )
  :effect
   (and
      (not (at ?truck ?loc-from))
      (at ?truck ?loc-to)
      (increase (electricity-used) (* (electricity-per-minute ?truck)
```

```
                                        (time-to-drive ?loc-from ?loc-to)))
 )
)

(:action WALK
  :parameters
   (?driver - driver
    ?loc-from - location
    ?loc-to - location)
  :precondition
   (and
      (at ?driver ?loc-from)
      (path ?loc-from ?loc-to)
   )
  :effect
   (and
      (not (at ?driver ?loc-from))
      (at ?driver ?loc-to)
      (increase (walked) (time-to-walk ?loc-from ?loc-to))
   )
)


)
```

7.8 Driverlog Metric

This domain is the same as Driverlog. The only difference is in problem files.

7.9 Production

This domain is available in Nor H. M. Radzi's PhD Thesis [50].

7.10 Bread

This domain is available in Nor H. M. Radzi's PhD Thesis [50].

Bibliography

[1] *PDDL4J pddl parser*, `http://sourceforge.net/projects/pdd4j/`.

[2] *VAL the plan validator*, http://www.inf.kcl.ac.uk/research/groups/PLANNING/index.php?option=com_content&view=article&id=70:val-the-plan-validator&catid=38:tools&Itemid=77, Version: 4.2.08.

[3] J. Bibai, P. Savéant, M. Schoenauer, and V. Vidal, *An Evolutionary Metaheuristic for Domain-Independent Satisficing Planning*, Proc. Int. Conf. on Automated Planning and Scheduling (ICAPS), AAAI Press, 2010, pp. 15–25.

[4] B. Bonet and H. Geffner, *Planning as heuristic search*, Artificial Intelligence **129** (2001), no. 1, 5–33.

[5] C. M. Fonseca and L. Paquete and M. Lopez-Ibanez, *An Improved Dimension-Sweep Algorithm for the Hypervolume Indicator*, Evolutionary Computation, 2006. CEC 2006. IEEE Congress on, 2006, pp. 1157 – 1163.

[6] W. M. Carlyle, J. W. Fowler, E. S. Gel, and K. Bosun, *Quantitative comparison of approximate solution sets for bi-criteria optimization problems**, Decision Sciences **34** (2003), no. 1, 63–82.

[7] T. M. Chan, *Klee's measure problem made easy*, 54th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2013, 26-29 October, 2013, Berkeley, CA, USA, 2013, pp. 410–419.

[8] A. Coles, M. Fox, D. Long, and A. Smith, *A hybrid relaxed planning graph–lp heuristic for numeric planning domains*, Proc. Int. Conf. on Automated Planning and Scheduling (ICAPS) (2008).

[9] A. J. Coles, A. I. Coles, M. Fox, and D. Long, *Forward-chaining partial-order planning*, Proc. Int. Conf. on Automated Planning and Scheduling (ICAPS), vol. 20, 2010.

[10] F. J. Damerau, *A technique for computer detection and correction of spelling errors*, Commun. ACM **7** (1964), no. 3, 171–176.

[11] I. Das and J. E. Dennis, *A closer look at drawbacks of minimizing weighted sums of objectives for pareto set generation in multicriteria optimization problems*, Structural optimization **14** (1997), 63–69.

[12] R. Dechter, I. Meiri, and J. Pearl, *Temporal constraint networks*, Artificial Intelligence **49** (1991), 61–95.

[13] M. B. Do, J. Benton, and S. Kambhampati, *Planning with goal utility dependencies*, Proc. Int. Conf. on Automated Planning and Scheduling (ICAPS) (2006), 23.

[14] M. B. Do and S. Kambhampati, *Sapa: A multi-objective metric temporal planner*, J. Artif. Intell. Res. **20** (2003), 155–194.

[15] M. Fox and D. Long, *Pddl+: Modelling continuous time-dependent effects*, Proc. 3rd International NASA Workshop on Planning and Scheduling for Space, 2002.

[16] ———, *PDDL 2.1: An Extension to PDDL for Expressing Temporal Planning Domains*, J. Artif. Int. Res. **20** (2003), 61–124.

[17] R. Fuentetaja, D. Borrajo, and C. Linares, *Improving relaxed planning graph heuristics for metric optimization*, In Proc. 2006 AAAI Workshop on Heuristic Search, Memory Based Heuristics and its Applications, 2006, pp. 79–86.

[18] _____, *A new approach to heuristic estimations for cost-based planning*, Proc. of 21st Int. FLAIRS Conference. FLAIRS'08, May 2008.

[19] _____, *A look-ahead b&b search for cost-based planning*, Proceedings of the Current topics in artificial intelligence, and 13th conference on Spanish association for artificial intelligence, Springer-Verlag, 2009, pp. 201–211.

[20] _____, *A unified view of cost-based heuristics*, Proceedings of Workshop on Heuristics for Domain-Independent Planning, ICAPS'09 (Thessaloniki, Greece), 2009, pp. 70–77.

[21] H. Geffner and P. Haslum, *Admissible heuristics for optimal planning*, Proc. Int. Conf. on AI Planning Systems (AIPS) (2000), 140–149.

[22] A. Gerevini and D. Long, *Preferences and soft constraints in pddl3*, ICAPS workshop on planning with preferences and soft constraints, 2006, pp. 46–53.

[23] A. Gerevini, A. Saetti, and I. Serina, *LPG-TD: a Fully Automated Planner for PDDL2.2 Domains*, Proc. Int. Conf. on Automated Planning and Scheduling (ICAPS) (2004).

[24] A. Gerevini and I. Serina, *Lpg: A planner based on local search for planning graphs with action costs.*, AIPS, vol. 2, 2002, pp. 281–290.

[25] M. E. Giuliano, R. Rager, and N. Ferdous, *Towards a Heuristic for Scheduling the James Webb Space Telescope*, Proc. Int. Conf. on Automated Planning and Scheduling (ICAPS) (2007), 160–167.

[26] M. Helmert, *The fast downward planning system.*, J. Artif. Intell. Res.(JAIR) **26** (2006), 191–246.

[27] M. Helmert and C. Domshlak, *Landmarks, critical paths and abstractions: What's the difference anyway?*, Int. Conf. on Automated Planning and Scheduling (ICAPS), 2009.

[28] J. Hoffmann, *Ff: The fast-forward planning system*, AI Magazine **22/3** (2001), 57–62.

[29] _____, *The Metric-FF planning system: Translating "ignoring delete lists" to numeric state variables*, J. Artif. Intell. Res.(JAIR) **20** (2003), 291–341.

[30] J. Hoffmann and B. Nebel, *The FF Planning System: Fast Plan Generation Through Heuristic Search*, J. Artif. Int. Res. **14** (2001), 253–302.

[31] C. L. Hwang, S. R. Paidy, K. Yoon, and A. S. M. Masud, *Mathematical programming with multiple objectives: A tutorial*, Computers & Operations Research **7** (1980), no. 1–2, 5 – 31.

[32] M. D. Johnston and M. Giuliano, *Multi-Objective Scheduling for the Cluster II Constellation*, International Workshop on Planning and Scheduling in Space (IWPSS) **6** (2011).

[33] E. Karpas and C. Domshlak, *Optimal search with inadmissible heuristics*, Int. Conf. on Automated Planning and Scheduling (ICAPS), 2012.

[34] E. Keyder and H. Geffner, *Heuristics for planning with action costs revisited*, 18th European Conference on Artificial Intelligence (ECAI), Patras, Greece, July 21-25, 2008, Proceedings, 2008, pp. 588–592.

[35] M. Khouadjia, M. Schoenauer, V. Vidal nad J. Dreo, and P. Saveant, *Pareto-Based Multi-objective AI Planning*, Int. Joint Conf. on Artificial Intelligence (IJCAI) **1** (2013).

[36] V. Klee, *Can the Measure of* $\bigcup[ai, bi]$ *be Computed in Less Than O(n log n) Steps*, The American Mathematical Monthly **84** (1977), 284–285.

[37] V. Levenshtein, *Binary coors capable or 'correcting deletions, insertions, and reversals*, Soviet Physics-Doklady, vol. 10, 1966.

[38] D. Long and M. Fox, *The 3rd International Planning Competition: Results and analysis*, J. Artif. Int. Res. **20** (2003), 1–59.

[39] R. T. Marler and J. S. Arora., *Survey of multi-objective optimization methods for engineering*, Structural and multidisciplinary optimization **26** (2004), no. 6, 369–395.

[40] D. McDermott, M. Ghallab, A. Howe, C. Knoblock, A. Ram, M. Veloso, D. Weld, and D. Wilkins, *PDDL The Planning Domain Definition Language*, Tech. report, 1998.

[41] A. Messac, *Physical programming-effective optimization for computational design*, AIAA journal **34** (1996), no. 1, 149–158.

[42] A. Messac, S. M. Gupta, and B. Akbulut, *Linear physical programming: A new approach to multiple objective optimization*, 1996.

[43] A. Messac and C. A. Mattson, *Normal Constraint Method with Guarantee of Even Representation of Complete Pareto Frontier*, AIAA Journal **42** (2004), 2101–2111.

[44] A. Messsac and C. A. Mattson, *Generating well-distributed sets of pareto points for engineering design using physical programming.*, Optimization and Engineering **3** (2002), no. 4, 431–450.

[45] T. Nguyen, M. Do, A. Gerevini, I. Serina, B. Srivastava, and S.Kambahampati, *Planning with Partial preference Models*, Technical Report (2011).

[46] T. A. Nguyen, M. B. Do, S. Kambhampati, and B. Srivastava, *Planning with Partial preference Models*, Int. Joint Conf. on Artificial Intelligence (IJCAI) **1** (2009), 1.

[47] S. Parkinson, A. Longstaff, A. Crampton, and P. Gregory, *The application of automated planning to machine tool calibration*, Proc. Int. Conf. on Automated Planning and Scheduling (ICAPS), 2012.

[48] ———, *Automated planning for multi-objective machine tool calibration: Optimising makespan and measurement uncertainty*, Proc. Int. Conf. on Automated Planning and Scheduling (ICAPS), 2014.

[49] J. Pearl and J. H. Kim, *Studies in semi-admissible heuristics*, Pattern Analysis and Machine Intelligence, IEEE Transactions on (1982), no. 4, 392–399.

[50] N. H. M. Radzi, *Multi-objective planning using linear programming*, Ph.D. thesis, University of Strathclyde, 2011.

[51] I. Refanidis and I. Vlahavas, *Grt: A domain independent heuristic for strips worlds based on greedy regression tables*, In Proc. ECP-99, Springer, 1999, pp. 346–358.

[52] ———, *The mo-grt system: Heuristic planning with multiple criteria*, In Proc. Workshop on Planning and Scheduling with Multiple Criteria. AIPS, 2002.

[53] S. Richter and M. Westphal, *The LAMA Planner: Guiding Cost-based Anytime Planning with Landmarks*, J. Artif. Int. Res. **39** (2010), 127–177.

[54] O. Sapena and E. Onaindia, *Handling numeric criteria in relaxed planning graphs*, Advances in Artificial Intelligence–IBERAMIA 2004, Springer, 2004, pp. 114–123.

[55] M. Sroka and D. Long, *Exploring Metric Sensitivity of Planners for Generation of Pareto Frontiers.*, Starting AI Research Symposium (STAIRS), 2012, pp. 306–317.

[56] ———, *Lpg based system for generation of pareto frontiers*, Plan SIG (2012).

[57] ———, *Building a Metric Sensitive Planner*, Plan SIG (2014).

[58] ———, *A cost-based relaxed planning graph heuristic for enhanced metric sensitivity*, STAIRS 2014 - Proceedings of the 7th European Starting AI Researcher Symposium, Prague, Czech Republic, August 18-22, 2014, 2014, pp. 270–279.

[59] S. V. Utyuzhnikov, P. Fantini, and M. D. Guenov., *A method for generating a well-distributed pareto set in nonlinear multiobjective optimization*, Journal of Computational and Applied Mathematics **223** (2009), no. 2, 820 – 841.

[60] V. Vidal, *A lookahead strategy for heuristic search planning.*, Proc. Int. Conf. on Automated Planning and Scheduling (ICAPS) (S. Zilberstein, J. Koehler, and S. Koenig, eds.), 2004, pp. 150–160.

[61] ———, *The yahsp planning system: Forward heuristic search with lookahead plans analysis*, 4th International Planning Competition, Citeseer, 2004, pp. 56–58.

[62] M. Ziadloo and S. S. Ghamsary, *A Framework to Evaluate Multi-Objective Optimization Algorithms in Multi-Agent Negotiations*, CIMSA 2009, Frontiers in Artificial Intelligence and Applications, 2009.

[63] B. D. Ziebart, J. A. Bagnell, and A. K. Dey, *Fast planning for dynamic preferences*, Proc. Int. Conf. on Automated Planning and Scheduling (ICAPS), 2008, pp. 412–419.

[64] S. Zionts and J. Wallenius, *An interactive programming method for solving the multiple criteria problem*, Management science **22** (1976), no. 6, 652–663.

[65] E. Zitzler, L. Thiele, M. Laumanns, C. M. Fonseca, and V. Grunert da Fonseca, *Performance assessment of multiobjective optimizers: An analysis and review*, IEEE Transactions on Evolutionary Computation **7** (2002), 117–132.