Improving the efficiency of the Pre-Optimization Plan Techniques

Asma Kilani

School of Computing and Engineering
University of Huddersfield

A thesis submitted to the University of Huddersfield in partial fulfillment of the requirements for the degree of
Master of Philosophy

January 2018

I would like to dedicate this thesis to my loving husband, parents, and kinds . . .

Declaration

Asma Kilani
January 2018

Acknowledgements

First of all, I would like to thank Allah for his many blessings.

I would like to thank my co-supervisor Dr. Lukas Chrpa who guided me throughout my journey. I gratefully acknowledge him invaluable help and endless support. I would like to thank my second supervisor Pro Lee McCluskey for his support and encouragement, and my main supervisor Dr.Hugh Osborne.

I want to express my heartfelt thank you to my husband, Khaled Hdiya for his support and help, and my lovley children Hiba, Mohamed, and Rana who have missed my good care for such a long time.

I want to thank my parents, Abd-Aslam Kilani and Somya Alamri, and my dear brothers and sisters for their constant love and support during my whole life.

Last, but not least, I want to thank my mother in law Mabroka Jobran and particularly to the spirit of my father in law, Mohamed Hdiya with all love and respect.

Abstract

Automated planning is an important research area of Artificial Intelligence (AI). In classical planning, which is a sub-area of automated planning, attention is given to ‘agile’ planning, i.e., solving planning problems as quickly as possible regardless of the quality of solution plans. Obtaining solutions quickly is important for real-time applications as well as in situations of imminent danger. Post-planning optimisation techniques for improving the quality of solution plans are a good option for improving poor quality plans. Since such techniques are run as post-processing, this avoids situations where there is a risk of not having solution plans in time. This thesis focuses on an important sub-area of post-planning optimisation; that is, on identifying and removing redundant actions from solution plans. In particular, this study extends the existing Action Elimination and Greedy Action Elimination algorithms by introduce two approaches to improve their efficiency. The AE and GAE algorithms are thereby developed into the UAIAE and UGAIAE systems respectively. The key to our approaches is based on optimise the process while keeping the same ‘elimination power’ (identifying and removing the same number of redundant actions). First approach improves the algorithms by considering situations where inverse actions are redundant, while the other identifies a subset of actions that cannot be present in any redundant actions set. This subset is named justified unique actions. The study’s approach to identifying this subset has been motivated by a promising heuristic approach called ‘landmarks’, which are facts or actions that cannot be eliminated to achieve the goal.

The approaches in this study have been empirically evaluated using several benchmark domains, as well as several planning engines that participated in the Agile track of the International Planning Competition 2014. In addition, they have been evaluated against state-of-the-art optimal and satisficing planners, as well as they are evaluated against a plan repair technique.

The methods of AE family can be understood as polynomial methods that improve the quality of a plan by removing redundant actions, or as tools to complement more sophisticated plan optimisation techniques.

Table of contents

List of figures

List of tables

Chapter 1

Introduction

This chapter gives an introduction to AI planning and to post-planning plan optimisation. It also presents the contributions of this work and describes the outline of this dissertation.

1.1 AI Planning

In normal English, planning can mean many different kinds of things, such as project planning, pension planning and urban planning. In Artificial Intelligence, AI Planning is the research area that studies the process of selecting and organising actions in order to achieve desired goals (Ghallab et al., 2004). AI planning is an important area for study as it directly contributes to the scientific and engineering goals of AI. The scientific goal of AI planning is to understand the principles that make intelligence behaver possible and the planning is an important component of rational behaver. The engineering goal of AI is to build autonomous intelligent machines (robots) in which planning engines are embedded in such a way that the control loop of the machine (robot) consists of sensing, planning and acting stages. However, AI planning engines generate plans or solutions to given planning problems which can be passed to an autonomous agent (robot), which can then execute these plans in order to achieve the desired goals.

Classical Planning is the simplest form of AI Planning. It is interested in finding the sequence of actions that transforms an initial state to goal state. This sequence is called a plan. *"Classical AI planning is concerned mainly with the generation of plans to achieve a set of pre-defined goals in situations where most relevant conditions in the outside world are known, and where the plan's success is not affected by changes in the outside world."* (Yang, 2012). In other words, the classical planning system relies on a number of restrictive assumptions (Ghallab et al., 2004), such as a finite number of states, complete knowledge about the state, deterministic actions, static state of the world that changes only when an action is applied, an attainment goal, sequential plans, and implicit time.

1.2 Motivation

In general, classical planning is known to be computationally intractable (PSPACE-complete) (Bylander, 1994; Erol et al., 1995). There are two different approaches to finding the solution to a planning problem. One is a satisficing approach that finds any solution, while the other is an optimal approach that finds the optimal solution with minimal total cost (typically minimal length). In some cases, these two approaches might differ in terms of complexity, as it has been proven that in many cases, finding an optimal solution is NP-hard, whereas finding any solution is tractable (i.e., solvable in polynomial time) (Helmert, 2003, 2006b). Many modern planning engines are ‘satisficing’; that is, they produce correct but not necessarily optimal solutions (the number of actions in plans may be higher than necessary). This allows the planner to be more efficient, which is particularly useful in real time situations when any correct plan produced is better than no plan. LPG (Gerevini et al., 2004), which performs a greedy local search on a planning graph, is a good example of the satisficing planner preference for obtaining a solution quickly rather than one of better quality. Other well-known satisficing planners, such as Fast Downward (Helmert, 2006a), LAMA (Richter and Westphal, 2010) and Mercury (Katz and Hoffmann, 2014) use an anytime approach; they attempt to quickly find an initial plan of possibly low quality, then use the remaining time to improve upon this plan. Moreover, sequential agile track planners, which were introduced in IPC2014 (Vallati et al., 2015) such as YAHSP3 (Vidal, 2014), Madagascar (Rintanen, 2014), PROBE and BFS(f) (Lipovetzky et al., 2014), focus on finding solutions to challenging problems and do not consider plan quality. In contrast to satisficing planning engines, optimal planning engines such as SymBA*-2 (Torralba et al., 2014a), GAMER (Edelkamp and Kissmann, 2008) and its extension cGamer(Kissmann et al., 2014), which are based on exploring binary decision diagrams, are focused on finding the best (shortest) plans. However, optimal planning is usually more time-consuming and therefore might be inappropriate for real-time applications.

Various methods are used to measure plan quality. The two most common methods for unit cost actions are the length of the plan (number of actions), and the execution time of the plan (makespan) if the actions can be executed in parallel. For non-uniform action cost, the total cost of the plan is defined as the sum of the cost of all actions. However, post-planning plan optimisation is a good option for improving poor quality plans generated by satisficing planners. It is planner-independent and run as post-processing step. In addition, some techniques optimise plans in polynomial time. Such process takes a valid plans and looks for opportunities to shorten them. This post-processing step is very useful when compromising between the speed of the planning process and the quality of solutions.

The motivation behind post-planning plan optimization techniques reflects situations in which there is a need to obtain a plan in a short time, for instance, when a robot is in imminent danger and must act quickly. However, when the plan is returned, there still may be some time to optimise it through post-planning analysis.

Different techniques have been proposed for post-planning plan optimization. These techniques can be classified into two categories:

- Pre-optimisation techniques identify and remove only redundant actions from plans in polynomial time (e.g., Balyo et al., 2014; Chrpa et al., 2012a; Nakhost and Müller, 2010). These are very important tools to complement more sophisticated plan optimisation techniques.

- More sophisticated techniques are more complex techniques that can find better (shorter) plans. Examples include a plan optimisation technique based on genetic programming (Westerberg and Levine, 2001); exploring state space around the plan in order to find shorter (more optimal) plans (Nakhost and Müller, 2010); replacing (sub)sequences of actions by shorter ones (Chrpa et al., 2012a; Estrem and Krebsbach, 2012); and decompiling a given plan into subplans and optimising each subplan locally (Siddiqui and Haslum, 2015; Siddiqui et al., 2013).

This thesis deals with pre-optimisation techniques that improve plan quality by removing redundant actions. The aim is to improve the efficiency of Action Elimination and Greedy Action Elimination algorithms, in order to provide a computationally easy method for determining redundant actions.

1.3 Contribution of the Thesis

This work provides an overview of existing post-planning plan optimisation techniques and presents the current techniques that improve a given plan in polynomial time. The main contribution, however, is extension of the Action Elimination (Nakhost and Müller, 2010) and Greedy Action Elimination (Balyo et al., 2014) algorithms that improve the quality of a plan by identifying and removing redundant actions by introducing two approaches to improve their efficiency (reducing the CPU-time) while keeping the same 'elimination power' (identifying and removing the same number of redundant actions). The first approach involves incorporating an Inverse Action Elimination algorithm feature (Chrpa et al., 2012a,b) into them and develop two new algorithms named AIAE and GAIAE respectively. The second approach involves developing a new algorithm to extract justified unique actions (actions that cannot be present in any redundant set) in a given plan and integrating this algorithm into AIAE and GAIAE, thereby developing two further algorithms named UAIAE and UGAIAE respectively (from the original AE and GAE).

The extraction of this subset of the actions is derived from a promising heuristic approach called 'landmarks'. The Landmarks are facts or actions that can not be eliminated to achieve the goal and can be achieved more than once and every solution plan has to achieve them at some point (Hoffmann et al., 2004). On the other hand, the justified unique actions are the subset of actions in a given plan that cannot be eliminated from the plan because they introduce unique facts that achieve the goal.

All the approaches in this thesis are accompanied by the necessary theoretical foundations. In addition, they are empirically evaluated using several benchmark domains and several planning engines that participated in the Agile track of the 2014[1] International Planning Competition. Furthermore the approaches are also evaluated against anytime planners, an optimal planner and a plan repair strategy.

1.4 Structure of the Thesis

This thesis is structured into seven chapters including the current one. The current chapter has briefly introduced the research problem, and the contributions. The reminder of the thesis is organised as follows:

- **Chapter Two** provides relevant background information with regard to the AI planning area. It starts with a brief history of AI planning and presents conceptual model of AI planning and its components. Subsequently, it covers classical planning representation, and classical planning techniques.

- **Chapter Three** reviews and analyses existing plan optimization techniques, providing a theoretical background in plan optimisation. In addition, it gives deep insight into pre-optimization algorithms.

- **Chapter Four** presents this study' approach to extending the pre-optimization algorithms, Action Elimination and Greedy Action Elimination. It provides some theoretical foundations for incorporating Inverse Action Elimination into them, and describes the new extended algorithms.

[1]https://helios.hud.ac.uk/scommv/IPC-14/

- **Chapter Five** deals with the approach to identifying the subset of the actions that cannot be removed from a plan. It provides the preliminaries on which this approach is based, as well as some theoretical foundations for identifying them, and describes a new algorithm that extracts this subset from the plan. Furthermore,it describes how exploit this subset to improve pre-optimistaion plan process and presents the implementation of this approach with the extended algorithms.
- **Chapter Six** presents different empirical evaluations based on several benchmark domains and several planning engines that participated in the Agile track of the International Planning Competition 2014. It provides the setting of the experiments, compares between the different classes of actions in the plan, evaluates the contributions of this study to Action Elimination and Greedy Action Elimination algorithms and provides a discussion of the results.
- **Chapter Seven** summaries the contributions of this thesis, and and discusses some interesting challenges and suggests some areas for future work.

Chapter 2

Background and Terminology

This chapter provides the background relating to AI planning. It covers the basic elements of planning and classical planning approaches.

2.1 An overview of AI Planning

AI planning became an active research area in the 1960s as a result of various attempts to create programs aiming to simulate human problem-solving abilities. One of the first such programs was General Problem Solver (Newell and Simon, 1963). The General Problem Solver applied actions that reduced the difference between an existing state and a goal state. Later, a new problem-solver known as STRIPS (Stanford Research Institute Problem Solver) was introduced (Fikes and Nilsson, 1971). In STRIPS, the states are represented as sets of propositions, whilst operators are represented by their effects and pre- and post-conditions, and the solution is a sequence of operators leading from the initial state to a goal state. In 1984, STRIPS was developed to become the planning component for controlling the Shakey robot(Nilsson, 1984).

Systems in the classical planning era (until the 1990s) applied methods including state-space or plan-space search, heuristics and hierarchical decomposition, among others. In 1998, the Planning Domain Definition Language (PDDL) was developed in preparation for the first International Planning Competition (McDermott et al., 1998). Since that time it has become the standard language for the planning community.

In the last two decades, researchers have successfully applied automated planning in many applications including space exploration, such as the Mars Rover(Estlin et al., 2003); manufacturing, such as the software to plan sheet-metal bending operations(Gupta et al., 1998); and games, such as Bridge Baron(Smith et al., 1998).

During recent years, automated planning has presented classical approaches that assess 'toy' problems, such as those applied in International Planning Competitions which simulate real-world problems but with numerous simplifications and assumptions. In this kind of problem, many aspects must be taken into consideration by the planner, such as time and resources; furthermore, the planner must support more expressing knowledge representations, plan in dynamic environments, etc. However, there are many issues of practical importance that can also be modelled as planning problems.

Modern approaches to classical planning almost always use a state-space heuristic search. In addition, a lot of approaches translate the planning problem into a SAT problem or into a SAS+ problem. There are also many portfolio approaches where the planner consists of a set of old planners.

2.1.1 Conceptual Model for Planning

A conceptual model is a simple theoretical device that describes the main components of a problem needing to be solved, and which helps in terms of gaining understanding and formalisation (Ghallab et al., 2004). The model shown in Figure 2.1, describes the interaction between three components: a state-transition system, a controller and a planner

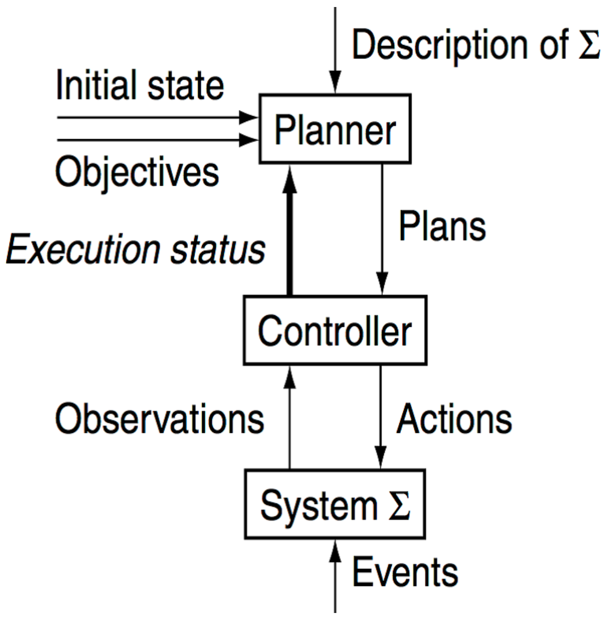


Fig. 2.1 Conceptual Model for Planning (Ghallab et al., 2004)

2.1.1.1 State- Transition System

A state transition system is also called a discrete-event system (Dean and Wellman, 1991). It is a formal model of the real-world system for which we want to create plans, and it deals only with the aspects that the planner needs to reason about(Ghallab et al., 2004). It is specified by a 4-tuple :$\sum$ = (S, A, E, γ), where:

- $S = \{s_0, s_1, s_2, \cdots\}$ is a finite set or recursively enumerable of states. These are all possible states the world can be in.
- $A = \{a_1, a_2, \cdots\}$ is a finite or recursively enumerable set of actions. The actions can be performed by the agent in order to modify the world.
- $E = \{e_1, e_2, \cdots\}$ is a finite or recursively enumerable set of events. Events, similarly to actions, can change the state of the world; however, the agent has no explicit control over them.
- $\gamma : S \times (A \cup E) \rightarrow 2^S$ is a state-transition function.γ takes two arguments as inputs: state of the world and actions or events. The result of applying the state transition function is another set of states (all possible states as result of applying actions or events).

This definition can be used in order to formally define some other concepts,such as applicability. If *a* is an action (*a* ∈ A), and γ(s, a) is not empty, then an action *a* is applicable in state *s*. Applying *a* in *s* will take the state transition system to some state in $\gamma(s, a)$.

State Transition System Example

Figure2.2 describes a state-transition system for a simple domain involving a container in a pile, a crane that can pick up and put down the container, and a robot that can carry the container and move it from one location to another. Here, the state transition system for this domain can be described as follows:

- $S = \{s_0, s_1, s_2, s_3, s_4, s_5\}$, the set of states;
- $A = \{move_1, move_2, put, take, load, unload\}$, the set of actions;
- $E = \{\}$, there are no events;
- γ is a set of states, but not necessarily a set of all possible states.

Moreover, each state transition leads to just one other state, i.e.,it is *deterministic*. For example, $\gamma(s_0, take) = s_1$, $\gamma(s_4, move2) = s_5$ and so on.

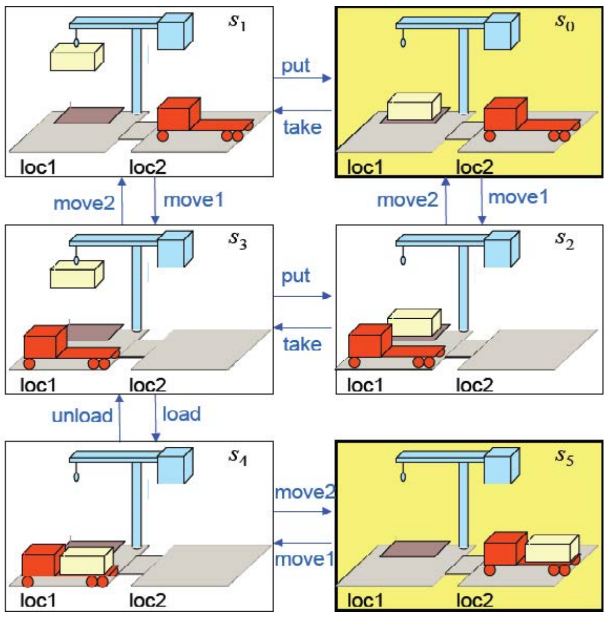


Fig. 2.2 A state- transition system for a simple domain involving a crane and a robot for transporting containers (Ghallab et al., 2004)

2.1.1.2 Planner

A planner is a solver that takes a planning problem as input and provides the plan or policy able to solve the planning problem. The planning problem includes a description of the system $\sum$, an initial situation and some objective (goal state, set of goal states, set of tasks, etc.). For example in Figure 2.2 the planning problem P consists of the initial state s_1, and a single goal state s_5, and the planner's output is the plan or policy that solves the planning problem. The plan is take, $move_1, load, move_2$, which is the sequence of actions. This plan will be passed on to the controller for execution.

2.1.1.3 Controller

The controller takes the plan generated by the planner and executes the actions in the plan,thereby changing the state of the system, such as in regard to the real world. The system is not only changed by the actions that are controlled by the controller; it is also changed because events may occur. In order for the controller to take appropriate actions, it usually needs to establish the current state of the system so that it receives observations from the system. The observation can be modelled as an observation function $\eta : S \rightarrow O$ that maps the state to a set of observations that can be made in the state. Thus, the controller takes the observation $O = \eta(s)$, where s is the current state as input.

As a consequence, the conceptual model is not very realistic as the real world in which we may wish to execute a plan is different from the description of the state transition system given to the planner. The reason for this is the description given to the planner is an abstraction where some real world attributes cannot be (properly) modelled. The controller and the planner must be powerful in order to overcome the difference between $\sum$ and the real world.

2.1.2 Types of Planners

Automated planning systems are categorized into three classes, based on their capability to configure work in different planning domains (Nau, 2007): these classes are domain specific planners, domain configurable planners and domain independent planners.

2.1.2.1 Domain-Specific Planners

These planners are made for specific planning domains, and are difficult to generalise to other planning domains. Importantly, they use specific forms of representation and techniques appropriate to each problem. This category includes most planners that have been developed in practical applications such as Mars exploration, sheet-metal bending, playing bridge, etc.

2.1.2.2 Domain-Independent Planners

Domain-independent planners accept domain models and problem specifications as inputs and provide plans as outputs. They deal with problems without specific knowledge of the domain. In essence, they work in any planning domain. In practical, it is very difficult to make domain independent planners work well in all possible planning domains. They need restrictions in terms of the type of planning domain.

2.1.2.3 Domain-Configurable Planners

Domain-configurable and domain-specific planners benefit from domain-specific knowledge, which serves to restrict the search to a small part of the search space. In a domain-configurable planner,the planning engine is domain independent engine, but a collection of domain-specific knowledge (domain description) is given as input to the planner. In other words, domain-configurable planners are based on domain-independent planners, which can compute and exploit additional knowledge about planning domains. Consequently, the planning engine is able to be configured to work in another problem domain by giving it a new domain description. Existing domain-configurable planners can be divided into two main types:

hierarchical task network (HTN) planners such as O-Plan (Tate et al., 1994) and Shop2(Nau et al., 2003), and control-rule planners such as Talplanner(Kvarnström and Doherty, 2000).

However, the domain-configurable planners are based on domain-independent planners, which can compute and exploit additional knowledge about planning domains.

2.2 Classical Planning Representation

The description of the planning problem is a very important input for any planning algorithm. Practically, this problem description cannot include a straightforward listing of all possible states and transitions. Such a listing would make a description extremely large, and the work required to generate would be greater than that involved in solving the planning problem. Instead of this, it is important to represent the planning problem in a way which makes it easy to compute the state and state transitions quickly.

Classical representation is a more expressive form two other representation schemes including set-theoretic and state-variable representation. It uses a concept originating from first order logic to generalise the set-theoretic representation. It represents the states as sets of logical atoms and represents the actions by planning operators.

Definition 1. *A planning operator is a 4-tuple $o = (name(o), pre(o), eff^-(o), eff^+(o))$, where:*

- *$name(o)$,the name of the operator, is expressed in the form $n(x_1,...x_2)$,where n is termed as an operator symbol, x_1, ...,x_k are all of the variable symbols that are appeared anywhere in o, and n is unique;*
- *$pre(o), eff^-(o) and eff^+(o)$ are sets of(unground) predicates representing the operator's precondition, negative and positive effects respectively.*

Definition 2. *An action is any grounded instance of planning operators. If a is an action and s is a state where $pre(a) \subseteq s$, then a is applicable to s, and the outcome of applying a to s is the state: $\gamma(s,a) = (s \setminus eff^-(a)) \cap eff^+(a)$.*

Definition 3. *Let L be a first-order language. The planning domain is a restricted state-transition system $\sum = (S, A, \gamma)$ such that:*

- *$S \subseteq 2^{allgroundatomsofL}$;*
- *A=all grounded instances of operators in O*
- *$\gamma(s,a) = (s \setminus eff^-(a)) \cap eff^+(a)$ if a is applicable to s;*

- *If $s \in S$, then for each action a is applicable to $s, \gamma(s,a) \in S$.*

Definition 4. *A classical planning problem is a triple $\Pi = (\sum, s_0, g)$ where:*

- *s_0, is the initial state;*
- *g is the goal(any set of ground predicates);*
- *$\{S_g = s \in S|$ s satisfy g $\}$.*

A *plan* is a sequence of actions. A plan *solves* a planning problem if a consecutive application of actions in the plan, starting in the initial state, results in a state where all goal atoms are present (the goal state). A plan solving a given planning problem is *optimal* if and only if there does not exist a plan solving the problem that is shorter (in terms of the number of actions in the action sequence). It should be noted that as well as sequential plans, there are also parallel plans, sequences of sets of independent actions, or partial-order plans.

2.2.1 Classical Planning Techniques

2.2.1.1 State Space Planning

In the case of state space planning, each node represents a state of the world, actions identify transitions between nodes, and a plan is a current path through the search space. They can be based on progression- a forward search from the initial state looking for the goal state- or they can be based on regression; in other words, a backward search from the goals towards the initial state. The search in these algorithms is not efficient without good heuristic function, which estimates the cost from a state to the goal. In the case of STRIPS planners, the cost of each action is 1; thus, the cost is the number of actions. The basic idea of planning heuristic is to consider the actions' effects and the goals that must be achieved, as well as the number of actions needed to achieve all goals. Finding the exact number is intractable; however most of the time it is possible to establish reasonable estimates without too much computation. In heuristic search planners, such as those by (Bonet and Geffner, 1999), the heuristic estimates are extracted by analysing the planning problem automatically, the heuristic then combined with standard search algorithms to achieve a search in the state space. FF planner(Hoffmann and Nebel, 2001) depends on a forward search in the state space, and is guided by a heuristic that estimates goal distances by ignoring negative effects.

The ability to derive an admissible heuristic that does not overestimate is also possible; this could be used with an A* search to find optimal solutions. Two approaches can be used: the first is to derive a relaxed problem from the given problem specification by ignoring negative effects; the second approach is the subgoal independence assumption, where the cost of solving a conjunction of subgoals is approximated by the sum of the costs of solving each subgoals independently.

2.2.1.2 Planning as Heuristic Search

Heuristic search is regarded as promising approach by state-of-the-art classical planners such as((Hoffmann and Nebel, 2001);(Gerevini and Serina, 2002);(Hoffmann, 2003) and (Richter and Westphal, 2010)). They depend on heuristic evaluation function that estimates the cost of the solution or the distance from any given state to the goal to guide the search towards goal states. Although the heuristic search is efficient in finding solutions to great planning problems, the solutions are not optimal due to non-admissible estimates or the search algorithm used. On the other hand, admissible heuristics are guaranteed to find optimal plans because they never overestimate the distance to the goal. Although finding the accurate number is NP-hard, it is possible to find feasible estimates most of the time.

Heuristic estimators are based on four categories: abstractions (e.g.,(Culberson and Schaeffer, 1998); (Helmert et al., 2014);(Katz and Domshlak, 2010)), delete relaxations (e. g., (Bonet and Geffner, 2001);(Hoffmann and Nebel, 2001);(Keyder and Geffner, 2008);(Katz et al., 2013)), critical paths ((Haslum and Geffner, 2000)), and landmarks (e. g., (Richter et al., 2008); (Karpas and Domshlak, 2009);(Helmert and Domshlak, 2009);(Keyder et al., 2010)). Currently,the landmarks category play an important role in raising the performance of satisfying planners.

2.2.2 Plan Space Planning

In plan-space planning,the search space is a set of partial plans; nodes represent partially-ordered plans and actions transform (refine) one partially-ordered plan into another. Plan space planning differs from state space planning not only in its search space but also in its definition of the solution plan. It uses a more general plan structure than a sequence of actions. The general strategy of delaying a choice during search is referred to as the least commitment strategy, which avoids adding to the partial plan any constraint that is not strictly needed for addressing the refinement purpose. Here, planning is considered as two separate operations: (1) the choice of actions,and (2) the ordering of the chosen actions to achieve the goal. A plan is defined as a set of planning operators together with ordering constraints and binding constraints. A partial plan can be defined as any subset of actions, precedence relations and causal links. The actions keep some useful part of the plan structure, with precedence relations between actions(e.g. action move-robot-AB must be performed before load-pack-B), and causal links are relations of the form: action 1 $\rightarrow (e)$ action 2, which means that action 1 has an effect *e* which is required by action 2. Plan space planners work back from the goal to achieve each sub-goal in the plan. They refine the partial plan by future ordering and constraining its actions or by adding new actions anywhere in the partial plan, as long as the constraints in plan can be satisfied.

2.2.3 Planning as Satisfiability

Planning as satisfiability is very successful approach in classical planning. It was proposed in 1992 by Kautz and Selman (Kautz et al., 1992). The main idea of planning with SAT is to transform the planning problem into a propositional satisfiability problem, using efficient SAT solver in order to extract the solution for the planning problem from a translated CNF formula. Some modern planners are based on SAT solver for example, Madagascar (Rintanen, 2014) and Freelunch (Balyo, 2014).

Chapter 3

Plan Optimization Background

This chapter presents an overview of state-of-the-art approaches for post-planning plan optimization. First all approaches in general are presented; thereafter the chapter focus on pre-optimization approaches that deal with identifying and removing redundant actions from a solution plan.

3.1 Existing Techniques for Post-Planning Plan Optimization

Several techniques have been proposed for post-planning plan optimization. For example,using genetic programming in post-planning plan optimization might provide some promising results (Westerberg and Levine, 2001). However, it is unclear whether such an approach is domain-independent (i.e. whether it is required to handle code optimization policies for each domain) and, moreover, the running time of such a method might be high. Another technique for plan improvement is used in the Aras system (Nakhost and Müller, 2010). This system takes any valid plan as input and improves it by iterating between two techniques. First,it applies an Action Elimination(AE) algorithm to remove redundant actions. Secondly,it creates a Plan Neighborhood Graph (PNG) using a breadth-first search and expanding a limited number of states (nodes) around each state along the plan, which is often a small subset of the original state space. Then it applies Dijsktra's algorithm to find a shorter path in the neighborhood graph, which might lead towards better, i.e. shorter plans. PNGS is extremely useful for local improvement of plan quality; although this approach might not work well if some 'optimizable' actions lie far from each other in the plan, in some permutations of the plan, actions can be adjacent.

AIRS (Estrem and Krebsbach, 2012) improves plans by identifying ‘optimizable’ subsequences of actions according to heuristic estimation, and attempts to find shorter (optimal) ones by utilising more expensive (optimal) planning techniques and eventually replacing the longer sequence with the shorter one. The heuristic estimation is used for estimating a distance between given pairs of states. If states seem to be closer than they are in the plan, then an optimal or near-optimal planner is used to re-plan between these states.

Other work (Chrpa et al., 2012b) proposes a method that explores plan structures based on analysing action dependencies and independencies (Chrpa and Barták, 2008) in order to identify redundant actions or non-optimal subplans. Firstly, the method identifies and removes all the actions upon which the goal is not dependent, which basically corresponds to backward justified actions (Fink and Yang, 1992). Secondly, all pairs of inverse actions reverting each other’s effects are checked for redundancy, which depends only on actions placed in between the pair of inverse actions, and these are eventually removed. An extension allowing grouped nested pairs of inverse actions to be removed, which covers situations when it is only possible to remove all the pairs together and not step by step, it has been introduced in (Chrpa et al., 2012a). Thirdly, the method identifies pairs of weakly adjacent actions, i.e., actions that can be adjacent in some permutations of the plan, and if possible replaces them with a single action. The weak adjacency of action is determined through use of the action independence relation, which allows swapping adjacent actions without affecting plans’ validity. An algorithm for determining weak adjacency of actions in plans has also been used for learning macro-operators (Chrpa, 2010). This approach has presented various positive aspects, such as being able to reasonably optimize plans in a short time. It is efficient in combination with fast satisfcing planners (e.g. LPG). Although this method is quite restricted to specific cases (e.g. inverse actions, non-optimal subplans of length two), it might be very useful as a preprocessing step to some other method (e.g. PNGS).

It is also worth mentioning an approach that optimises parallel plans (Balyo et al., 2012). This approach improves plans locally where (parallel) subplans of a pre-defined length k could possibly be replaced by shorter ones. After all the subplans are processed, k is incremented and the optimization process is performed until the length of subplans reaches (or exceeds) a given limit.

Recently, an approach for optimal planning with inadmissible heuristics has been proposed (Karpas and Domshlak, 2012). Shortcut rules are introduced, which are learnt and applied during the planning process; in concrete terms, they are used for deriving existential optimal landmarks and for removing redundant actions from partial plans. In general, shortcut rules can be used to replace some action sequences (totally or partially ordered) by shorter (or less expensive) action sequences without violating the correctness of plans.

These rules can be obtained from several sources, including learning them online, during the planning process, such as plan rewrite rules(Nedunuri et al., 2011). Plan quality optimisation via block decomposition (Siddiqui et al., 2013) is the most recent work in sophisticated plan optimization techniques. This method decomposes a given plan using plan deordering into blocks of partially ordered subplans, which are optimized autonomously using a bounded cost search and then substituted into the plan. Later this work has since been extended (Siddiqui and Haslum, 2015). The strength of this method lies in its ability to optimize subplans where actions may lie far from each other in a totally ordered plan. Another recent work are introduced by Balyo et al.,2014 adapts the action elimination algorithm reinvented by Fink and Yang 1992 and Nakhost and Mullure2010 in order to remove more expensive redundant actions. In addition, Balyo et al.,2014 propose two new algorithms based on partial maximum satisfiability (PMaxSAT) and weight partial maximum satisfiability (WPMaxSAT) solving to remove the set of redundant actions with a maximum possible total cost.

3.2 Pre-Optimization Approaches

This section will focus on a subset of state-of-the-art approaches for post-planning plan optimization. It will present pre-optimization techniques that are interested in identifying and removing redundant actions from the plan in polynomial time.

3.2.1 Preliminaries

Naturally, actions in sequential plans influence each other in some way. For instance, actions achieve atoms of other actions that need them as their precondition. These relationships between actions are called *dependencies* (Chrpa et al., 2012b), which can be defined as follows:

Definition 5. *let $\langle a_1, ...a_n \rangle$ be an ordered sequence of actions. Action a_j is **directly dependent** on action a_i (denoted as $a_i \longrightarrow a_j$) iff $i < j$, $(eff^+(a_i) \cap pre(a_j)) \neq \emptyset$ and $(eff^+(a_i) \cap pre(a_j)) \nsubseteq \cup_{t=i+1}^{j-1} eff^+(a_t)$.*

Solution plans are thus structured in a way which can be exploited in order to gather additional knowledge that might be useful on many different occasions, for instance, in generating macro-operators that can speed up the planning process (Chrpa, 2010; Chrpa et al., 2014). The de-ordering of solution plans into blocks has been shown to be useful for optimising these plans (Siddiqui et al., 2013). Analysing the structure of solution plans can identify actions that if removed from the solution plan, the plan remains a valid. Such actions are called *redundant* actions (Chrpa et al., 2012a). The formal definition is as follows:

Definition 6. *Let* Π *be a planning problem and* π *its solution plan. The actions* $a_{x_1},\dots,a_{x_k} \in \pi$ *are* ***redundant*** *in* π *(or form a set of redundant actions in* π*) iff* $\pi' = \pi \setminus \{a_{x_1},\dots,a_{x_k}\}$ *is a solution plan of* Π.

Identifying the largest set of redundant actions is NP-complete (Nakhost and Müller, 2010). This is not a very desirable result. On the other hand, a lot of redundant actions can still be identified and removed in polynomial time (Balyo et al., 2014). In literature (Nakhost and Müller, 2010), a plan procured by removing redundant actions called reduction(π'), and the minimal reduction is a lowest-cost plan that can be obtained by removing redundant actions.

Definition 7. *Let* π' *is subsequence of* π*,* π' *is* ***a reduction*** *of the plan* π*, symbolized by* (π,π') ***iff*** π' *is also a plan for* Π

Definition 8. *let* $\pi = \langle a_1,\dots..a_n\rangle$ *is solution plan to planning problem. The* ***cost*** *of* π *is the sum of action costs, cost(*π*)=*$\sum_{i=1}^{n} f(a_i)$

Definition 9. π' *is called* ***minimal reduction*** *of* π *iff for every* π'' *such that reduct(*π',π''*),*$cost(\pi') \leq cost(\pi'')$

3.2.2 Plan Justifications

The notion of plan justification was introduced by Fink and Yang 1992. A justified plan is a plan that does not include any actions which are not needed to achieve the goal. In their work, Fink and Yang **?** define various kinds of plan justifications and present algorithms to find justified plans.

3.2.2.1 Backward Justified Plan

The idea of a backward justified plan is derived from partial order plans which represent a time precedence relation between operators. The backward justified plan is a solution plan in which every action introduces an important atom to achieve the goal. The actions in justified plans are called *backward justified* actions. The idea behind backward justified actions can be formalised in the following definition:

Definition 10. *Let* $\pi = \langle a_1,\dots.a_n\rangle$ *be a plan solution to planning problem* Π*. An action* a_i *is called* ***backward justified*** *iff* $\exists p \in eff(a_i)$ *such that* a_i *provides* p *either for the goal or for another backward justified action.*

Finding a backward justified subplan is not hard, but it may possibly contain some redundant actions. For example, the action a_i is in some cases not truly justified and can be removed from the plan without violating the correctness of the plan. Such a case may occur when an other action provides the same atom that is important to achieve the goal.

3.2.2.2 Well Justified Plan

A well Justified plan is a plan that does not contain any action can be removed without violating the correctness of the plan. The actions in a well justified plan are named well justified actions, the idea behind well justified action can be formalised in the following definition:

Definition 11. *Let $\pi = \langle a_1, ...a_n \rangle$ be a plan solution to planning problem Π. An action $a_i \in \pi$ is called **well-justified** iff $\exists p \in eff(a_i)$ such that a_i provides p either for some action or for the goal, and p does not hold before a_i.*

A well justified plan does not include any action that is not necessary to achieve the goal, and it stronger than backward justified plan.

3.2.2.3 Perfectly Justified Plan

Although, well Justified plans cannot contain unnecessary actions, they may still contain unnecessary groups of actions. This means that while no single action may be removed from the plan, a subset of its actions can possibly be removed together. However,the plan is called perfectly justified if no subset of its actions possibly removed from the plan. The idea behind a perfect plan can be formalised as follow:

Definition 12. *A valid plan is called perfectly justified iff it does not have any shorter subplan that achieve the goal.*

Although perfect justification is powerful than other justifications,and perfectly justified plan does not has redundant actions, it cannot found in polynomial time. It has been proven that finding perfectly justified plan is NP-complete (Fink and Yang, 1992).

3.2.2.4 Greedily Justified Plan

As mentioned before, it is NP-hard to find a perfectly justified plan. The greedy justification task seeks to find a nearly perfectly justified plan by removing a subset of actions that are not *greedily justified*. The following definition describes greedily justified action formally:

Definition 13. *Let $\pi = \langle a_1, ...a_n \rangle$ be a plan solution to planning task Π, $a_i \in \pi$, and $depend = (a_{i1}, ..., a_{ik})$ is a set of actions that depends on a_i and $depend \in \pi$ .The action a_i is called **greedily justified** iff $\pi_{\prime} = \pi \backslash (a_i \cup depend)$ is not solution for* Π.

Ti can be seen that a plan is greedily justified if all its actions are greedily justified. In addition, every well justified plan is backward justified, and every greedily justified plan is well justified. The algorithms that describe how to find justified plans are presented in more details in the work (Fink and Yang, 1992).

3.2.3 Action Elimination Algorithm

Action Elimination(AE) (Nakhost and Müller, 2010) is an algorithm which identifies and removes some redundant actions from solution plans of classical planning problems (see Algorithm 3). AE is derived from the concept of greedily justified actions (Fink and Yang, 1992) descried above. AE iteratively checks whether actions are greedily justified and if they are not, such actions and their dependents can be removed from the plan without compromising its validity. In particular, an intermediate step of the main loop a_i action is marked for removal. Then, an attempt is made to apply actions starting from a_{i+1} to the end of the plan. If an action is not applicable, then it is marked for removal and its effects are ignored (it is not applied). After reaching the end of the plan, it can be verified whether the goal is satisfied. If so, the marked actions are redundant and can be removed.

3.2.4 Inverse Action Elimination Algorithm

In many planning domain models, it is the case that planning operators reverse each other's effects (e.g., in the BlocksWorld domain, they are pickup(?x) and putdown(?x) operators that reverse each other's effects). Hence, solution plans might consist of actions that reverse each other's effects, in other words, *inverse actions*, in other words. The formal definition is as follows:

Definition 14. *The actions a and a' in plan are **inverse** iff for every state s such that a is applicable in s, it is the case that $\gamma(s, \langle a, a' \rangle) \subseteq s$.*

It should be noted that the definition considers actions as inverse even if their consecutive application in some state results in its sub-state. Since negative preconditions (i.e., not having an atom present in a state) are not considered, all actions that can be applicable in a sub-state of a state are also applicable in that state. Such assumption does not invalidate the rest of the plan if the inverse actions (a and a') in a solution plan are redundant (and removed). The

Algorithm 1: AE

```
Input  : $s_0$, plan $\pi = (a_1, .....a_n)$,and goal state G
Output : $\pi'$
$s \leftarrow s_0$;
i$\leftarrow$ 1;
while $i < n$ do
    mark $a_i$;
    $s' \leftarrow s$;
    for $j \leftarrow i+1$ to $n$ do
        if $applicable(a_j, s')$ then
            $s' \leftarrow$ apply($s', a_j$);
        else
            mark $a_j$;
        end
        if $s'$ does not satisfy $G$ then
            unmark all actions;
            s $\leftarrow apply(s', a_i)$;
        else
            $\pi' = \pi \setminus$ markedactions;
        end
    end
    i $\leftarrow$ i+1
end
return $\pi'$ ;
```

Inverse Action Elimination (IAE) (Chrpa et al., 2012a,b) algorithm identifies and removes pairs of redundant inverse actions from the plan(see Algorithm 2). The main idea behind the IAE algorithm is formalised in the following proposition:

Proposition 1. *Let $\pi = \langle a_1, \dots, a_n \rangle$ be a solution plan of some planning problem. Let $a_i, a_j \in \pi$ $(i < j)$ be inverse actions. If there is no action a_k, where $(i < k < j)$, such that $\mathit{eff}^+(a_i) \cap pre(a_k) \neq \emptyset$ or $\mathit{eff}^-(a_k) \cap \mathit{eff}^+(a_j) \neq \emptyset$, then a_i and a_j are redundant in π.*

Proof. See (Chrpa et al., 2012a). □

Although IAE algorithm is specific and does not cover all possibly redundant actions in the plan, it is beneficial in some domains.

Algorithm 2: IAE

```
   Input  : plan $\pi = (a_1, .....a_n)$
   Output : $\pi^‘$
 1 s$'$ ← initial state ;
 2 while $i < n$ do
 3     for $j \leftarrow i+1$ to $n$ do
 4         if $a_i$ and $a_j$ are inverse then
 5             Rinv ← false ;
 6             for $k \leftarrow i+1$ to $j-1$ do
 7                 if ($a_k$ is dependent on $a_i$) $\vee \mathit{eff}^-(a_k) \cap \mathit{eff}^+(a_j) \neq \emptyset$ then
 8                     Rinv ← True;
 9                     break;
10                 end
11             end
12         end
13     end
14     if $\neg$ Rinv then
15         mark both $a_i$ and $a_j$
16     end
17     i← i+1;
18     $\pi' = \pi \setminus$ markedactions;
19 end
20 return $\pi'$;
```

3.2.5 Greedy Action Elimination

Action elimination(see Algorithm 1) removes a set of redundant actions as soon as they are discovered and ignores the cost of them. Greedy action elimination (Balyo et al., 2014) is modified version of action elimination that first identifies all the sets of redundant actions and subsequently eliminates the redundant set with the highest cost(sum of the actions 'costs) and repeats this process until no more set of redundant actions are detected. Details of greedy action elimination are given in Algorithm 3. Initially, it calls *AECost* a function that identifies the subsets of redundant actions and returns the most costly set of redundant actions; then, the greedy action elimination removes this set. This process is reiterated until no such set is found. Greedy action elimination does not guarantee to remove all redundant actions in the plan. It outperforms AE in some cases (Balyo et al., 2014).

3.2.6 SAT-Based Algorithms

SAT Based Algorithms (Balyo and Chrpa, 2014; Balyo et al., 2014) are dependent on SAT encoding for the problem of detecting redundant actions in the plan. One algorithm is called Minimal Length Reduction(MLR) and this aims to eliminate the maximum subset of redundant actions from the plan and without considering the action cost. It contracts the optimized plan using a satisfying task that obtained from Partial Maximum Satisfiability(PMaxSAT) solver. Another algorithm is called Minimal Reduction(MR) and it is similar to the first one. One difference, however is that MR depends on a Weighted Maximum Satisfiability(WPMaxSAT) solver. MR removes the actions with a maximal total cost where all actions have a nunit cost.

Although such methods guarantee to eliminate the best set of redundant actions in the plan they are NP-complete, so they need exponential time to run (in general).

3.3 Comparison of Techniques of AE family to Related Work

Techniques of AE family can be understood as a complement tools to the state-of-the-art plan optimisation techniques such as PNGS (Nakhost and Müller, 2010), AIRS (Estrem and Krebsbach, 2012) and genetic programing (Westerberg and Levine, 2001). They identify and remove redundant actions from plans (without compromising their validity), which can often be done in a short time. Since other methods are based on more sophisticated, often time-consuming techniques that can optimise suboptimal parts of plans by finding their "better replacements", they can benefit by "pre-optimisation" from the AE family. In addition, the AE family can efficiently identify redundant actions that are placed far from each other in a plan; while the other methods are useful for local enhancement, they do not take into account a plan's structure.

Algorithm 3: GAE

```
Algorithm GAE($\pi$)
    n← lenght of the plan
    repeat
        BestCost=0
        Cost=0
        BestSet=AECost($\pi$)
        if BestSet is not empty then
            $\pi' = \pi \setminus$ BestSet
        end
    until BestSet is empty
    return $\pi'$
Function AECost($\pi$)
    $s \leftarrow s_0$
    for $i \leftarrow 0$ to $i < n$ do
        mark $a_i$
        $s' \leftarrow s$
        for $j \leftarrow i+1$ to $n$ do
            if applicable($a_j, s'$) then
                $s' \leftarrow$ apply($s', a_j$)
            else
                mark $a_j$
            end
            if $s'$ does not satisfy $G$ then
                Unmark all actions
                s $\leftarrow apply(s', a_i)$
            else
                cost=cost of marked actions
                if $cost >= Bestcost$ then
                    Bestcost=cost
                    Best-Redundant-Set=Marked Actions
                end
            end
        end
    end
    return Best-Redundant-set
```

Chapter 4

Improving the Time for Redundancy Checking in AE and GAE Algorithms

4.1 Identifying Redundant Actions

Although finding and removing all redundant actions from a plan is NP-hard, there are several polynomial algorithms that can remove redundant actions such as Action Elimination (Fink and Yang, 1992; Nakhost and Müller, 2010), Greedy Action Elimination (Balyo et al., 2014) and Inverse Action Elimination (Chrpa et al., 2012a). They all have a basic structure which can be described as follows:

- Identify the subset of actions in a plan based on specific criteria.
- Mark these actions to be removed.
- Check the plan' validity.
- If the rest of the plan remains valid, remove marked actions from the plan.

Since every action in the solution plan is tested, this process results in time complexity of AE and GAE algorithms. For example, in the Action Elimination Algorithm the time complexity is $O(n^2 p)$, and in Greedy Action Elimination (Balyo et al., 2014) it is O(n^3p), where n=$|\pi|$ and p is the maximum number of preconditions of the actions. On the other hand, Inverse Action Elimination does not remove all possible redundant actions, but it provides a computationally easy way of identifying a specific set of the redundant actions in a given plan and cares only about actions in between a pair of inverse actions (see Proposition 1, Chapter 3), and does not have to go through the rest of the plan to check the redundancy. Consider a simple Blockword task shown in figure 4.1, there are three blocks referred to as

A, *B* and *C*, such that the initial state is *on(A, B)* and *ontable(C)*, and the goal situation is *on (A, B)* and *on(B,C)*. If a solution plan likes this: *unstack(A, B), stack(A,C), unstack(A,C), putdown(A),pickup(B), stack(B, C), pickup(A), stack(A, B)* , in such a case, *stack(A,C)* and *unstack(A,C)* are inverse redundant actions and can be removed from the plan. If AE and GAE algorithms are applied to identify and remove these actions, they will check all the actions before removing both of them. On the other hand, IAE will identify and remove them without checking the rest of the plan.

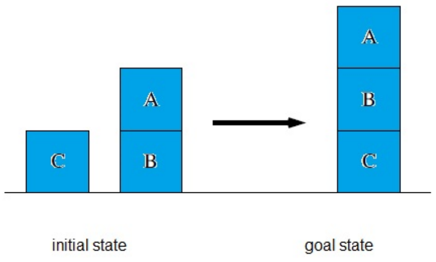


Fig. 4.1 An Example Blockword Task 1

Lemma 1. *Let $\pi = \langle a_1, ...a_n\rangle$ be a plan solution to planning problem Π, a_i, a_j are inverse redundant actions in the plan, k the number of actions placed between them ,and l the number of the actions placed after a_j. Then:*

- *IAE only needs o(k) steps to identify them.*
- *AE needs O(k+l) steps to identify them .*
- *GAE needs O((k+l)n).*

Proof. It can observed that since the IAE algorithm just identifies and removes a_i and a_j from the plan by checking the redundancy of actions between them only, it needs *o(k)* steps to check if there is any action between them which depends on a_i. On the other hand, AE and GAE check all the actions between a_i and a_j and all the actions that are located after a_j if they depend on a_i. In this case, in one iteration that contains a_i and a_j, AE needs *O(k+l)* steps, and GAE needs *O((k+l)n)* because it repeats that n times.

$\square$

From Lemma 1 it can be concluded that there is possibility of saving some CPU-time in *AE* and *GAE* in some cases where two actions in one iteration are redundant and inverse. This can optimise the process while keeping the same 'elimination power' in terms of identifying and removing the same number of redundant actions. To determine the redundancy of pairs of inverse actions only actions placed in between the pair of inverse actions need to be investigated.

4.2 Extended Algorithms

Following a vision based on incorporating the IAE algorithm feature into AE and GAE ones, Two algorithms named AIAE and GAIAE have developed respectively in order to improve the CPU-time of AE and GAE by modifying the step of identifying redundant actions (Kilani, 2015).

4.2.1 AIAE Algorithm

AIAE is a standard to incorporate the Inverse Action Elimination Algorithm feature into the Action Elimination one *(IAE+AE)*. The Action Elimination algorithm (Nakhost and Müller, 2010) identifies and removes some redundant actions from the plan by iteratively checking actions for 'relaxed greedy justification' and, if removing the involved actions does not affect the validity of the plan, the plan is updated (optimized); otherwise, the removed actions are restored, and the plan remains unchanged (for more details see Chapter 3). Although AE is strong and fast, in some cases it cannot identify certain kinds of redundant actions and does not care about the interactions between actions in the plan. The AIAE algorithm modifies AE one by considering situation where two actions in one iteration are inverse. This states under what conditions a pair of inverse actions in a solution plan is redundant(see Proposition 1 , Chapter 3).The idea behind the AIAE algorithm formulated in the following proposition:

Proposition 2. *Let $\pi = \langle a_1, ...a_n \rangle$ be a plan solution to planning problem Π, a_i and a_j are inverse redundant actions in π. Then the steps required to identify a_i and a_j as redundant actions by AIAE are fewer than the steps required to identify them by AE.*

Proof. Following Lemma 1, assume that a_i and a_j are redundant inverse actions in plan π, k is the number of actions placed between them ,and l is the number of actions placed after a_j. In one iteration that contains a_i and a_j, AE needs *o(k+l)* steps to identify them as redundant actions in the plan . On the other hand , AIAE only needs *o(k)* steps.

□

Details of AIAE are given in Algorithm 4. The algorithm is divided into two phases. First, it marks every action in the plan and all the action that depend on it (line 5 to line 16). In the next phase it checks the validity of the plan without the marked subset. If the plan is correct, it removes this subset of actions otherwise; it unmarks this subset (line 16 to line 22) and goes for the next iteration, continuing thus until the final action in the plan.

In the main loop it marks a_i as an action to remove and tries to apply the actions that are located after it. If an action is applicable it applies this action in the state and considers its effects. Otherwise, it marks this action to be remove as its preconditions are not satisfied. After reaching the end of the plan, it tests whether the current state satisfies the goal. If so, it removes the marked actions as they are redundant. Otherwise, the marked actions are not redundant, and cannot be removed.

AIAE modifies AE by adding an intermediate steps in lines 12,13 and 14. Before moving on to test a next a_j action, AIAE checks if a_i, and a_j are inverse with no actions marked between them(actions dependent on a_i). If so (proposition 1 in Chapter 3 holds), AIAE will mark aj without checking the rest of the actions in the plan. It will break j loop and remove a_i and a_j(this is the main difference between AE and AIAE). This will reduce the CPU-time spent on iterations that contain inverse actions, and will be more efficient when the inverse actions are laying close to each other in the plan.

4.2.2 GAIAE Algorithm

GAIAE is a standard to incorporate the Inverse Action Elimination Algorithm feature into the Greedy Action Elimination one *(IAE+GAE)*.The Greedy Action Elimination algorithm which presented in (Balyo et al., 2014) and described in Chapter 3, is an extension to the Action Elimination one which is expected to give better quality results at the cost of increased CPU-time. It identifies all sets of redundant actions and chooses the most expensive set to remove, and repeats that until no sets of redundant actions are found. Due to the repetition of the main loops n times, the time complexity of the algorithm is high. The GAIAE algorithm extends the GAE one by considering situations where two actions in one iteration are redundant actions in the same AIAE situation, with respect to maintaining the quality of optimised

Algorithm 4: AIAE

```
1  Input $s_0$, plan $\pi = (a_1, .....a_n)$,and goal state g
   **Output :** $\pi'$
2  $s \leftarrow s_0$
3  i$\leftarrow$ 1
4  while $i < n$ do
5      mark $a_i$
6      $s' \leftarrow s$
7      for $j \leftarrow i+1$ to $n$ do
8          if applicable($a_j, s'$) then
9              $s' \leftarrow$ apply($s', a_j$)
10         else
11             mark $a_j$;
12             if $a_i$ and $a_j$ are inverse and proposition 2 holds then
13                 remove ai and $a_j$ from the plan
14                 break
15             end
16         end
17         if $s'$ does not satisfy g then
18             unmark all actions
19             s $\leftarrow apply(s', a_i)$
20         else
21             $\pi' = \pi \backslash$ markedactions
22         end
23     end
24     i $\leftarrow$ i+1
25 end
26 return $\pi'$
```

plans while needing (generally) less CPU-time. The idea behind the GAIAE algorithm is formulated in the following proposition:

Proposition 3. *Let $\pi = \langle a_1, ...a_n \rangle$ be a plan solution to planning problem Π, a_i and a_j are redundant inverse actions in π. Then the steps required to identify a_i and a_j as redundant actions by GAIAE are fewer than the steps required to identify them by GAE.*

Proof. Following the Lemma 1, assume that a_i and a_j are redundant inverse actions in plan π, k is the number of actions placed between them, and l is the number of actions placed after a_j. In one iteration that contains a_i and a_j, GAIAE needs *o(k+l)* steps to identify them as redundant actions in one i iteration that is repeated n. On the other hand , AIAE only needs *o(k)* steps.

$\square$

Details of GAIAE are given in Algorithm 5. In this thesis, the algorithm has been in a different way from our original one (Balyo et al., 2014). GAIAE relies on an *AECost* function that identifies the subsets of redundant actions and returns the most costly set of redundant actions; the cost of set is computed by considering that every action has one unit cost. Firstly, AECost function takes an initial state, goal and plan as input and returns the best set of redundant actions, that containing the greater number of inverse actions. If this set is not empty, GAIAE removes this set from the plan. This process is reiterated until no such set is found.

4.2.3 AECost Function

The *AECost* function is the same as AE, GAE, and AIAE algorithms(lines: 2 to 18) in its ability to identify the subset of redundant actions. The feature of incorporation of IAE is simulated on lines 11 and 12 where if pairs of actions are redundant inverse actions(Proposition 1 Chapter 3 holdS), the j loop will be broken without checking the rest of the actions in the plan (this is the difference between GAE and GAIAE). If a temporal state satisfies the goal, then *AECost* will computes the cost of the marked actions and compare it with the best cost found until now, *if(cost $>=$ Bestcost).* If it is a best cost, *AECost* will identify the marked action set as the best redundant set, and do that in every i iteration. When all actions have been tested *AECost* will return the redundant set with the highest number of actions.

Algorithm 5: GAIAE

```
Algorithm GreedyAIAE($\pi$,$s_0$,$g$)
    n← length of the plan
    repeat
        BestCost←0
        BestIndex←0
        BestSet← AECost($s_0, g, \pi,$)
        if Bestset is not empty then
            $\pi' \leftarrow \backslash Bestset$
        end
    until BestSet is empty
    return $\pi'$
Function AECost($s_0, g, \pi$)
    $s \leftarrow s_0$
    for $i \leftarrow 0$ to $i < n$ do
        mark $a_i$
        $s' \leftarrow s$
        for $j \leftarrow i+1$ to $n$ do
            if applicable($a_j, s'$) then
                $s' \leftarrow$ apply($s', a_j$)
            else
                mark $a_j$
                if $a_i$ and $a_j$ are inverse and (proposition 1 holds) then
                    goto 20
                end
            end
        end
        if $s'$ does not satisfy $g$ then
            Unmark all actions
            s $\leftarrow apply(s', a_i)$
        else
            cost← cost of marked actions
            if $cost >= Bestcost$ then
                Bestcost ← cost
                BestRedundantSet ← MarkedActions
            end
        end
    end
    return BestRedundantSet
```

4.2.4 Correctness

Proposition 4. *Let $\pi = (a_1, \ldots, a_n)$ be a solution plan of some planning problem. Let A_{AE} be a set of actions that are removed from π by the AE algorithm and A_{AIAE} be a set of actions that are removed from π by the AIAE algorithm. Then, $A_{AE} = A_{AIAE}$.*

Proof. It can be observed that AE and AIAE differ only in the following situation. After an action a_i is marked for removal, and later on an action a_j is marked for removal too, then if a_i and a_j are inverse and redundant (the condition of Proposition1, Chapter3 holds), AIAE in contrast to AE, does not continue exploring the rest of the plan (after a_j) and proceeds directly to removing these actions. Proposition 1 (see Chapter 3); however, implies that actions applied after a_j remain applicable and their application results in a goal state(see 2). AE thus does not mark and remove any other action than a_i and a_j. □

Proposition 5. *Let $\pi = (a_1, \ldots, a_n)$ be a solution plan of some planning problem. Let A_{GAE} be a set of actions that are removed from π by the GAE algorithm and A_{GAIAE} be a set of actions that are removed from π by the GAIAE algorithm. Then, $A_{GAE} = A_{GAIAE}$.*

Proof. The proof is straightforward and similar to proof Proposition 3. GAE and GAIAE are only different in the following case. After an action a_i is marked for removal and later on an action a_j is marked for removal too, then if a_i and a_j are inverse and redundant (the condition of Proposition 1 in Chapter 3 holds), GAIAE does not continue exploring the rest of the plan (after a_j) (see Proposition 3) and proceeds directly to compute their cost without going through the rest of the plan.

□

Lemma 2. *Let Π be a planning problem π be a solution plan, $\pi\prime_{IAE}$ is a valid plan reduction produced by IAE, and $\pi\prime_{AE}$ is a valid plan reduction produced by AE, $\pi\prime_{AIAE}$, is a plan reduction produced by AIAE, $\pi\prime_{GAE}$, is a plan reduction produced by GAE and $\pi\prime_{GAIAE}$, is a plan reduction produced by GAIAE. Then $\pi\prime_{IAE}$,$\pi\prime and_{AE}$ and $\pi\prime_{AIAE}$ are* ***Identical*** *iff and only if all of the redundant set are pairs of inverse actions.*

Proof. Lemma 2 is a specific case when all the redundant actions are inverse. Let $A_{inverse}$ =((a_{i1}, a_{ji}),.......(a_{ik}, a_{jk})) be a set of inverse actions in a solution plan, and all of the pairs are redundant. Accordion to on all a theoretical background above if one algorithm identifies this subset as redundant actions, then AE, GAE, AIAE, GAIAE and will do as will. □

4.3 Summary

This chapter has described how the existing Action Elimination and Greedy Action Elimination algorithms were extended by considering situations where inverse actions are redundant. The extended algorithms were accompanied by the necessary theoretical foundations and empirically evaluated using several benchmark domains and several planning engines that participated in the Agile track of the International Planning Competition 2014 (see Chapter 6). They are very useful when the domain has a lot of inverse redundant actions and this will reduce the CPU-time of AE and GAE when inverse actions are close to each other in the plan.

Chapter 5

Justified Unique Actions and Pre-Optimization Techniques

This chapter demonstrates two key aspects of the present work: First, it introduces a new approach to identifying the subset of actions in a solution plan that cannot be presented at any set of redundant actions. Secondly, it describes how to exploit this approach to improve the efficiency of extended algorithms that we presented in chapter 4.

5.1 Introduction

Actions in a solution plan are classified into two categories. The first type are redundant actions that can safely be removed from the plan (for further detail see Chapter 3). The second type are non-redundant actions that cannot be removed from the plan. If we have a solution plan for planning problem, we can identify which actions that may be redundant or non-redundant. In the literature (Fink and Yang, 1992) non-redundant actions are called justified actions.

Current pre-optimisation techniques seek to improve plans by identifying a subset of actions to remove and testing the validity of the plan without this subset; if the plan remains valid, the subset is removed. The approach adopted in this work is different. Firstly, the subset of actions that can not be removed from the plan, or which cannot be present in a set of redundant actions, is identified; then, a subset of redundant actions is identified and removed. There are Some several issues to be considered here: (1) What criteria should be used to identify this subset? (2) How can this subset be exploited? (3) What is the expected improvements for this subset?.

5.2 Inspiration from Landmarks

In STRIPS planning,a state is a finite set of ground predicates or facts that represent the world. A planning problem is specified through a planning domain, an initial state and a set of goal facts. An initial state is a finite set of ground predicates or facts that represent the state of the problem, a goal is a conjunction of positive ground predicates or facts, and a state satisfies a goal if the state contains all of the conjunct facts in the goal. Actions are defined by action schema that consist of three parts: action name, preconditions and effects. Preconditions must hold before the action can be executed, and effects describe how the action changes the state of the world when it is executed.

Facts in planning are provided by the initial state or by actions that are introduced as effects. These facts are either preconditions for other actions or as the goal state facts. In planning literature, facts or subgoals that must appear at some point in every solution plan are called 'landmarks' (Hoffmann et al., 2004; Keyder et al., 2010; Porteous and Cresswell, 2002; Porteous et al., 2001; Richter et al., 2008). To illustrate this, it is helpful to consider a simple Nomystery problem, depicted in Figure 5.1 , with six locations l_0, l_1, l_2, l_3, l_4 *and* l_5, one trackt_0, and six packages $p_0, p_1, p_2, p_3, p_4 and p_5$. In the initial state,[(at p_0 l_2),(at p_1 l_2),(at p_2 l_1),(at p_3 l_3),(at p_4 l_3),(at p_5 l_5)]. The goal is to have the packages at their respective destinations, ((at p_0 l_3), (at p_1 l_4), (at p_2 l_3), (at p_3 l_2), (at p_4 l_2), (at p_5 l_4)) and initially these do not hold. For the goal to be achieved (loaded p_4 t_0 l_3), and *(loaded p3 t0 l3)* must occur at some point and thus they are landmarks for this task. In addition, goal facts are landmarks, and thus (at p_3 l_2) and (at p_4 l_2) are landmarks as well. In the following definition, terminology from Hoffmann et al. (2004) is adopted to represent fact landmarks.

Definition 15. *Given planning problem* $\Pi =$ *(A,I,G). A fact F is landmark iff for all* $\pi = \langle a_1, \ldots\ldots, a_n \rangle$, $G \subseteq$ *Result(I,*π*):F* $\in$*Result(I,(*a_1*,.....,*a_i*)) for some* $0 \le i \le n$.

From the fact landmarks, a specific subset of fact landmarks can be derived in order to identify a specific subset of actions that cannot be removed from a plan solution. The extraction of this subset is motived by the fact that the goal may contains unique facts achieved by one action within the plan's execution. Before this subset is identified, some terms need to be formally described.

Definition 16. *Let* $\pi = \langle a_1, \ldots\ldots, a_n \rangle$ *be a solution plan for planning task (A,I,G), F is positive fact, and* $G \subseteq$ *Result(I,*π*):F* $\in$ *Result((I,(*a_1*,.....,*a_n*)). F is called a* ***unique fact*** *iff F is present in the effects of only one action and* $F \notin s_0$.

Definition 17. *Let* $\pi = <a_1, a_2, \ldots\ldots a_n>$ *be solution plan to a planning problem. An action* a_i *is called a* ***Unique Action*** *in* π *iff it achieves some fact or facts which are not present in any action effects and also in the initial state.*

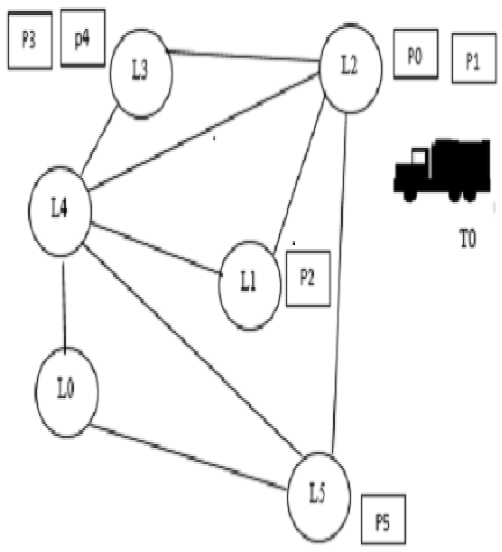


Fig. 5.1 An Example Nomystery Task

Definition 18. *Let $\pi =< a_1, a_2, ...a_n >$ be solution plan to planning problem Π, F is a Unique fact, and $F \in eff^{+}a_i$, and $F \in$ goal facts. Then F is called **goal unique fact**, and a_i is called a **goal unique action** that introduces F to the goal.*

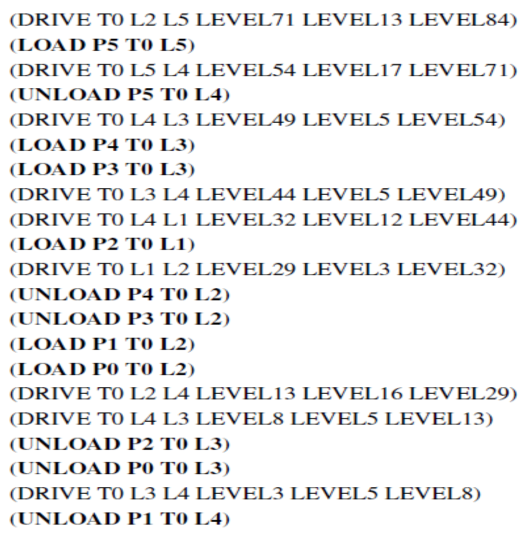
The above definition shows that if the plans contains some unique actions that introduce unique facts to a goal. Given a plan like that shown in Figure 5.2, according to the above definitions,(unload P_2 T_0 L_3) and (unload P_0 T_0 L_3) are goal unique actions, and (at p_0 l_3) and (at p_2 l_3) are goal unique facts.

The difference between landmarks and unique facts is that unique facts depend on a given plan and are achieved only once, landmarks can be achieved more than once and every solution plan has to achieve them at some point. For a goal facts, every unique fact is a landmark, and every landmark is not necessarily unique.

Proposition 6. *let $\pi =< a_1, a_2, ...a_n >$ be solution plan to planning task Π, a_i is a unique goal action. Then $\pi \backslash a_i$ is not solution to the planning problem.*

Proof. The proof can be shown in the following way. From the above example, action (unload P_2 T_0 L_3) introduces (at p_2 l_3) which is necessary to achieve the goal and (at p_2 l_3) is unique fact (see Definition 18). That means no other actions in the plan can provide this fact, and the goal cannot be achieved if (at p_2 l_3) is not achieved, therefor (unload P_2 T_0 L_3) can not be redundant in a solution plan. □

Clearly, a unique goal action can imply an other subset of actions that introduce unique preconditions to it. The following definition describes this relation formally:



```
(DRIVE T0 L2 L5 LEVEL71 LEVEL13 LEVEL84)
(LOAD P5 T0 L5)
(DRIVE T0 L5 L4 LEVEL54 LEVEL17 LEVEL71)
(UNLOAD P5 T0 L4)
(DRIVE T0 L4 L3 LEVEL49 LEVEL5 LEVEL54)
(LOAD P4 T0 L3)
(LOAD P3 T0 L3)
(DRIVE T0 L3 L4 LEVEL44 LEVEL5 LEVEL49)
(DRIVE T0 L4 L1 LEVEL32 LEVEL12 LEVEL44)
(LOAD P2 T0 L1)
(DRIVE T0 L1 L2 LEVEL29 LEVEL3 LEVEL32)
(UNLOAD P4 T0 L2)
(UNLOAD P3 T0 L2)
(LOAD P1 T0 L2)
(LOAD P0 T0 L2)
(DRIVE T0 L2 L4 LEVEL13 LEVEL16 LEVEL29)
(DRIVE T0 L4 L3 LEVEL8 LEVEL5 LEVEL13)
(UNLOAD P2 T0 L3)
(UNLOAD P0 T0 L3)
(DRIVE T0 L3 L4 LEVEL3 LEVEL5 LEVEL8)
(UNLOAD P1 T0 L4)
```

Fig. 5.2 A Plan solution for Nomystery task

Definition 19. *let $\pi = \langle a_1, ...a_n \rangle$ be a solution plan, let $a_i \in \pi$ and a_i be unique goal action,$p \in eff^{+}a_j$ and j<i, p is a unique, and $p \in pre(a_i)$. Then a_j is called a* ***related unique action****.*

From the above example,(load p_0 t_0 l_2) is a related unique action to goal unique action (unload P_2 t_0 l_3) that introduces unique fact(loaded p_0 t_0).

Proposition 7. *let $\pi =< a_1, a_2, ...a_n >$ be a solution plan to planning problem Π, a_j is a Unique Related Unique Action. Then $\pi \backslash a_j$ is not solution to the planning problem.*

Proof. Action a_j introduces unique fact p that is precondition to a_i and no other action can provide this precondition. This means that if a_j is removed from the plan, p can not be provided and a_i cannot be applied. Therefor, the goal cannot be achieved, and a_j cannot be redundant.

□

Definition 20. *let $A^* \subset \pi$ be unique goal actions in π, and $A' \subset \pi$ be related unique actions then $A^* \cup A'$ is called a* ***Justified Unique Actions****.*

Justified Unique Actions name is derived from the term Greedily Justified Action (Fink and Yang, 1992), an action which, if it and the actions that depend on are removed from the plan, causes the plan to become invalid (see Chapter 3).

From above theoretical foundations, following corollary can be concluded:

Corollary 1. *let $\pi =< a_1, a_2, ...a_n >$ be solution plan to planning task Π, $A^{ju} \subseteq \pi$ is a set of Justified Unique Actions in π. Then A^{ju} subset of actions that must appear in every plan reduction.*

Proof. Follow the proof of Proposition6and Proposition 7. A^{ju} cannot be redundant in a solution plan. Then when we try to optimise a plan by any plan optimization method such as AE or GAE we cannot eliminate this subset because of a plan reduction must have this subset to achieve a goal.

□

5.3 Extraction of Justified Unique Actions from Solution Plan

A pre-process that extract justified unique actions in a solution plan has been developed. It consists of two phases. In the first phase, backward search is performed to find a subset of actions that introduce unique facts to the goal. In the second phase, backward search is performed to identify the subset of actions that introduce unique preconditions to the unique actions identified in the first phase.

The pre-process is detailed in Algorithm 6. Firstly, the algorithm identifies first candidate facts that include goal facts and removes all facts that already provided by the initial state. After that, it starts from the last action in the plan. For every action, it test whether this action introduces any unique facts to goal. Unique Candidates facts are extracted using the *UniqueFact* function, which takes the set of facts and the action' position in the plan as input. For every goal fact present in the action effects, the function test whether it has yet been identified. If so, it adds an attribute to the goal fact (action position in the plan). Otherwise, it removes this fact from the unique facts set. After identifying all the unique facts in the goal, the algorithm identifies unique goal actions by their positions and add them to the Justified Unique Action(JUA) set.

In the next phase,the JUA identifies second unique candidate facts which include preconditions of unique goal actions and removes all facts already provided in the initial state. For every action not introduced in the unique goal action set, it tests whether an action achieves any unique facts by using *UniqueFact* function. Then, it identifies the related unique actions by their position in the plan and adds them to the Justified Unique Actions set. Finally, it return a set of Justified Unique Actions.

Algorithm 6: JUA

```
1  Algorithm JustifiedUniqueActions($\pi$,$\Pi$,$g$,$s_0$)
2      $Ufacts \leftarrow g \backslash s_0$ // initialise unique facts set and remove the facts
        that already in an initial state
3      UPrec$\leftarrow$ {}              // initialise unique preconditions set
4      $i \leftarrow n$                  // n=length of the plan
5      while $i \geq 1$ do
6          UniqueFact(Ufacts,i)          // identify unique goal facts
7          i$\leftarrow$i-1
8      end
9      foreach $fact \in Ufacts$ do
10         j$\leftarrow$ get attribute of fact
11         add $a_j$ to JuniqueActions
12         $UPrec \leftarrow UPrec \cup perc(a_j)$    // identify candidate precondition
            facts
13         $UPrec \leftarrow UPre \setminus s_0$     // remove the facts that already in an
            initial state
14     end
15     i$\leftarrow$ 1
16     while $i \leq n$ do
17         foreach $a_i \notin$ JuniqeActions do
18             UniqueFact( UPrec, $a_i$)     // identify unique preconditions
19         end
20         i$\leftarrow$ i+1
21     end
22     foreach $fact \in UPrec$ do
23         j$\leftarrow$ get an attribute of fact
24         add $a_j$ to JuniqueActions
25     end
1      Function UniqueFact($Facts, i$)
2          foreach $fact \in eff^{+}(a_i) and fact \in Facts$ do
3              if $fact$ is undefined in Facts then
4                  set attribute i to fact
5              else
6                  remove fact from Facts
7              end
8          end
9      return Justified Unique Actions
```

5.4 Improving Pre-Optimization Plan Process

Improving a pre-optimization plan process is based on identifying a subset of the actions that cannot be removed from the plan (justified unique actions). A key goal of this approach is to develop a complementary tools to more Sophisticated plan optimization techniques. Figure 5.3 shows a the structure of these process. It structures the pre-optimization plan process into three main stages, including (1) extracting a justified unique actions set, (2) identifying a subset of redundant actions by considering situations where two inverse actions are redundant; and (3) removing redundant actions. The main difference from previous methods is that, instead of checking every action in the plan and its dependents for redundancy, it firstly identifies justified unique actions (actions that cannot be present in any redundant action set) and, before going to mark any action, it tests whether this a action is non-redundant in a given plan. This will reduce the effort of checking the redundancy of every actions in a plan, and reduce CPU-time since process of extracting justified unique actions is in polynomial time.

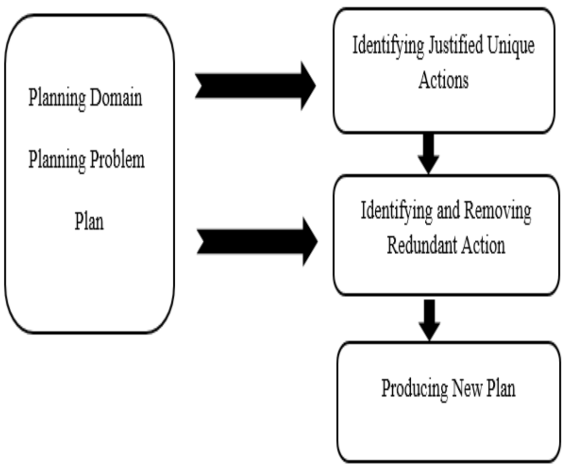

Planning Domain
Planning Problem
Plan
Identifying Justified Unique Actions
Identifying and Removing Redundant Action
Producing New Plan

Fig. 5.3 A New Structure for Pre-Optimisation Plan Process

5.4.1 Implementation of a New Algorithms

Through this structurer two new plan optimization methods has been developed. The algorithms implemented in Chapter 4 (AIAE and GAIAE) have been extended, and new methods are UAIAE and UGAIAE respectively. Details of UAIAE are given in Algorithm 7. The main difference between AIAE and UAIAE is that before determining whether any action in a given plan can be removed in every i iteration, it checks whether this action is a *justified unique action*; if so, it applies this action and continues to test the next actions in a plan. otherwise, it checks every action in j iteration if it also *justified unique action*. If so it continues to test next action in j iteration. Otherwise; it marks this action. To avoid repetition, U(GAIAE) one follows the same steps that UAIAE uses to identify redundant actions, considering action cost as GAIAE does.

5.5 Summary

This chapter has introduced and defined the concept of a justified unique actions set in a solution plan. It has presented a polynomial algorithm to extract this subset of actions. Implementation of the algorithm has accompanied by necessary theoretical foundations. In addition, This algorithm feature has been exploited to introduce a framework for the pre-optimization plan process. This is framework has been simulated with the extended algorithms that presented in Chapter 4. The new algorithms have been adapted using different approaches to work together in one algorithm, this enabling them to work as an efficient complementary tools to more sophisticated plan optimisation technique. They have been empirically evaluated using several benchmark domains and several planning engines that participated in the Agile track of the International Planning Competition 2014(see Chapter 6).

Algorithm 7: UAIAE

```
Input $s_0$, plan $\pi = (a_1, .....a_n)$,and goal state g
Output : $\pi'$
$s \leftarrow s_0$
$J_{UA} \leftarrow justifieduniqueactions$
i$\leftarrow$ 1
while $i < n$ do
    if $a_i \in J_{UA}$ then
        s $\leftarrow apply(s, a_i)$
        i $\leftarrow$ i+1
        continue
    end
    mark $a_i$
    $s' \leftarrow s$
    for $j \leftarrow i + 1$ to $n$ do
        if applicable($a_j, s'$) then
            $s' \leftarrow$ apply($s', a_j$)
        else
            mark $a_j$;
            if $a_i$ and $a_j$ are inverse and proposition 2 holds then
                remove $ai$ and $a_j$ from the plan
                break
            end
        end
        if $s'$ does not satisfies $g$ then
            Unmark all actions
            s $\leftarrow apply(s, a_i)$
        else
            $\pi' = \pi \backslash$ markedactions
        end
    end
    i $\leftarrow$ i+1
end
return $\pi'$
```

Chapter 6

Empirical Results

This chapter presents the experimental results of this work regarding improvement to the pre-optimization techniques AE and GAE. In addition, it presents the evaluation of improved techniques against state-of-the -art optimal and satisficing planners, and plan repair strategy.

6.1 Experimental Settings

- To test the efficiency of introduced approaches, Inverse Action Elimination(IAE), Action Elimination(AE), Greedy Action Elimination(GAE), and their extensions AIAE and GAIAE were implemented. In addition, Justified Unique Action (JUA) algorithm was implemented, and integrated into AIAE, and GAIAE.
- For benchmarks, domains from the sequential staisficing track of the 2011 and 2014 IPCs were tested. However, because the introduced techniques does not support some of PDDL features such as conditional effects, a limited number of IPC2014 domains, such as Barman, Floortile, Thoughtful and Transport were tested.
- To obtain the initial plans, five state-of-the-art agile track planners from the International Planning Competition(IPC2014) were used. Theses were, Yahsp3 (Vidal, 2014); Madagascar(Rintanen, 2014); Probe and Bfs(f) (Lipovetzky et al., 2014); and Jasper (Xie et al., 2014). These planners focus on finding solutions to challenging problems but do not consider plan quality. The planners had a time limit of five minutes per problem to find a solution plan.
- To evaluate the quality of the plans optimised by introduced methods against optimal plans, three top performers from IPC2014, SymbA*-2 (Torralba et al., 2014a); SPM&S

(Torralba et al., 2014b); and RIDA (Franco et al., 2014) were selected. The planners had a time limit of 30 minutes per problem to find a solution plan.

- To evaluate the performance of introduced methods against the incremental approaches for improving the quality of the plans, two satisficing planners, the winner of both the 2008 and 2011 IPC competitions LAMA (Richter and Westphal, 2010), and the runner-up planner of sequential satisfcing track in the IPC2014 Mercury (Katz and Hoffmann, 2014). Again, the planners had a time limit of 30 minutes per problem to find a solution plan.
- To evaluate the performance of introduced methods against plan repair strategy, LPG (Gerevini et al., 2003) were used to repair invalidated plans.

6.2 Experiments Requirements

- All methods are implemented in C++.
- All methods support typed STRIPS representation in PDDL.
- All the experiments were run on a 3.6 Ghz CPU with 16.0 GB of RAM.

6.3 Experiment 1: Performance of the Planners

This experiment compared the performance of three optimal planners and seven satisfycing planners. 180 problems were tested from IPC2011 domains, while from IPC2014 domains, 80 problems were tested. Figures 6.1 and 6.2 illustrate the number of problems per domain solved by the ten planners. It can be seen that agile planners were able to solve many problems in each domain except Madagascar planner. It was unable to solve problems in the Barman, Elevators, Parking, Sokoban and Transport domains. The anytime planners were able to solve most of the problems in all domains, but the optimal planners were unable to solve problems of Barman, Elevators and Transport domains.

6.4 Experiment 2: Evaluation of Incorporation of IAE Feature into AE and GAE Algorithms

The cumulative results (results of all problems considered in particular domains and planners) are presented in Tables 6.1 and 6.2. Four plan optimisation techniques, AE, AIAE, GAE

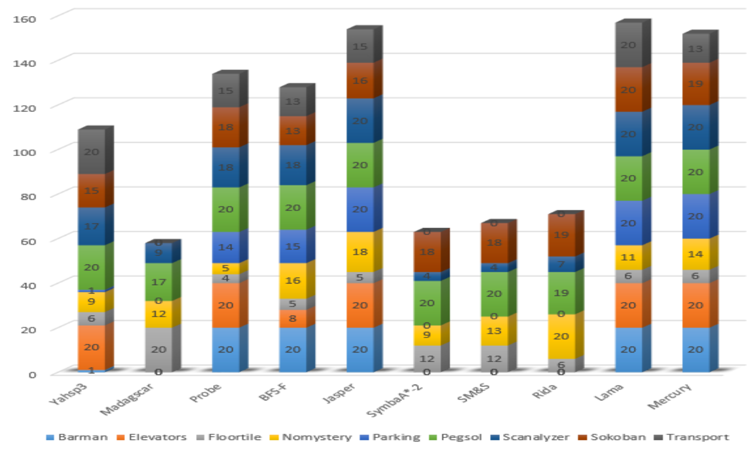


Fig. 6.1 Number of problems solved in IPC11 by different planners

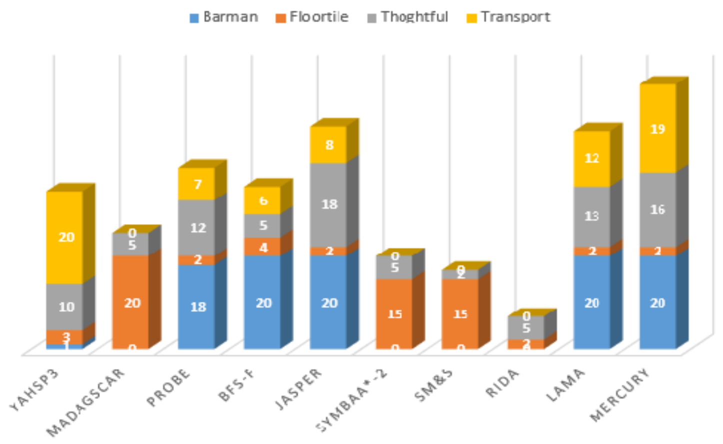


Fig. 6.2 Number of problems solved in IPC14 by different planners

and GAIAE, were compared with each other. For each of them, the total plan length Ω and the total run time $\Diamond$ to optimise the plans were measured. For the main comparison (see propositions 1 and 2 in Chapter 4), the IAE algorithm was used to find inverse redundant actions in a plan. For IAE, $\sum$ is the total number of inverse actions and $\Diamond$ the total run time to find them. For more detailed results see Appendix A.

- **Inverse Action Elimination (IAE)** algorithm is very fast compared with other algorithms. It identifies a specific set of redundant actions (pairs of inverse actions). For the IPC2011 domains, it was noted that most of the domains containing redundant actions, also contained redundant inverse actions, except for the Scanlyzer and Sokoban domains. In addition, in some domains where the number of redundant actions was high, the number of redundant inverse actions was high as well, for example in the Elevators, Transport and Barman domains. In the IPC2014 domains, all plans contained inverse actions except for those generated by Probe and Bfs-f for the Thoughtful domain. There was a high number of redundant inverse actions in plans generated by Yahsp3 for the Transport domain, and in Jasper plans for the Barman domain, the number was also high. An interesting result observed for the Parking domain was that all the redundant actions were inverse.

- **Action Elimination (AE)** algorithm removes the highest number of redundant actions in a very short time, except for in the Transport and Sokoban domains, where it took a long time. Among the IPC2011 domains, it was noted that there were no redundant actions in the Pegsol domain for any plans generated by any of the tested planners. Furthermore, among plans generated by the Probe and Bfs planners, there were no redundant actions in the Nomystery and Parking domains. On the other hand, all other domains contained redundant actions and the number of these was very high, especially in plans generated by Yahsp3 for the Elevators, Transport and Sokoban domains; in plans generated by Madagascar for the Floortile domain, and in plans generated by Jasper for the Barman domain. With regard to the IPC14 domains, it was observed that all domains contained redundant actions and the total number of these was very high in plans generated by Yahsp3 for the Transport domain and in plans generated by Jasper for the Barman domain.

- **AIAE** algorithm removes the same number of redundant actions as AE because it is an extended version of AE, but just considers the situations where redundant actions are inverse in order to reduce the CPU-time of AE (see propositions in Chapter 4).

- **Greedy Action Elimination (GAE)** algorithm generally removed the same number of redundant actions that AE and AIAE removed and the run time was not much increased in most cases, except for plans in the Transport and Elevators domains. The highest run time was, in most instances, due to the repetition of the process n times (n is number of the actions in the plan).

- **GAIAE** algorithm removes the same number of redundant actions as GAE. It is an extended version of GAE, but just considers situations where redundant actions are inverse in order to reduce the CPU-time of GAE (see proposition 2 in Chapter 4).

Generally, the total run time of AIAE and GAIAE algorithms is less than that of AE and GAE algorithms. The best overall results were achieved in plans generated by Yahsp3, especially in the Transport, Elevators and the Sokoban domains where the number of inverse actions was high.

6.5 Experiment3: Justified Unique Actions and Redundant Actions

This experiment demonstrates the relationship between justified unique actions and redundant actions in a plan solution. To calculate the number of redundant actions AE algorithm was used. For justified unique actions (actions that can not be eliminated to achieve the goal),the JUA algorithm was used (for more details, see chapter5).

The cumulative results of the experiment are displayed in Tables 6.3 and 6.4. With regard to the total number of justified unique actions, it was found that all plans in all domains contained justified unique actions except for some plans in the Sokoban domain. The total number of these, when compared with the total length of the plans, was generally high. For example, in the Nomystery domain, the total number was around 50 percent of the total plan length for all plans generated by the five planners.

However, looking at the total number of redundant actions and the total number of justified unique actions, an interesting result can be observed. In general, when the number of justified unique actions is high, the total number of redundant actions is low, especially in the Scanalyzer, Nomystery, Pegsol, Parking, Floortile and Thoughtful domains. Detailed results per problem are presented in Appendix A.

6.5.1 Experiments 4: Justified Unique Actions and Improving Pre-Optimization Plan Process

This experiment evaluates introduced approach to optimise plans by adding a pre-process that identify justified unique actions (non-redundant). The extended algorithms AIAE and

Table 6.1 Results of experiments on the plans for IPC11 domains found by the planners Yahsp3, Madagascar, Probe, Bfs-f, and Jasper. The column 'Plans' contains the number (Num) of the plans and the total length of them (T.Length). The column IAE contains the total number of the redundant inverse actions $\sum$ in the plans and the total run time in seconds $\Diamond$ required to find them. The following columns contains the total length of optimised plans Ω and the total time in seconds $\Diamond$ required for optimization process for the four evaluated algorithms.

Planner	Domain	Plans		IAE		AE		AIAE		GAE		GAIAE	
		Num	T.Length	$\sum$	$\Diamond$	Ω	$\Diamond$	Ω	$\Diamond$	Ω	$\Diamond$	Ω	$\Diamond$
yahsp3	Barman	1	235	84	153	0.1	137	0.6	137	0.4	137	20	137
	Floortile	6	303	32	0.06	268	0.3	268	0.3	268	1	268	0.9
	Nomystery	9	275	0	0.2	275	3	275	3	275	5	275	5
	Parking	2	137	4	0.1	73	0.1	73	0.1	73	0.2	73	0.2
	Pegsol	20	603	0	0.1	603	0.6	603	0.6	603	0.7	603	0.7
	Scanalyzer	17	2278	0	0.4	1898	6	1898	5	1898	61	1898	56
	Transport	12	3432	417	1	2693	102	2693	93	2734	1325	2731	1440
Madagascar	Floortile	18	1663	172	0.2	1452	10	1452	7	1452	94	1452	89
	Nomystery	12	381	24	5	354	75	354	68	354	165	345	158
	Pegsol	17	479	0	0.1	479	0.6	479	0.6	479	0.4	479	0.5
	Scanlyzer	5	166	9	0.01	142	0.2	142	0.2	142	0.2	142	0.3
Probe	Barman	18	2780	38	0.8	2732	7	2732	7	2732	16	2732	15
	Elevators	16	4078	40	3	3958	267	3958	266	3958	1592	3958	1658
	Floortile	4	181	6	0	172	0.05	172	0.0.7	172	0.4	172	0.4
	Nomystery	4	114	0	2	114	21	144	21	114	23	114	22
	Parking	11	1580	14	0.6	1553	2	1553	2	1553	3	1553	3
	Pegsol	20	565	0	0.2	565	0.7	565	0.8	565	0.8	565	0.8
	Scanalyzer	17	576	0	0.1	566	0.6	566	0.6	566	0.6	566	0.6
	Sokoban	18	5074	0	3	4852	112	4852	112	4850	236	4850	236
	Transport	13	3218	68	1	2955	94	2955	87	2949	507	2949	503
Bfs-f	Barman	17	2423	48	3	2369	6	2369	6	2369	19	2367	20
	Elevators	5	814	30	0.3	779	23	779	22	779	139	779	126
	Floortile	6	291	10	0	281	0.3	281	0.3	281	0.8	281	0.7
	Nomystery	17	610	0	12	610	181	610	182	610	183	610	183
	Parking	18	1528	0	0.3	1528	1	1528	1	1528	2	1528	2
	Pegsol	20	603	0	0.064	601	0.5	603	0.5	601	0.6	603	0.7
	Scanalyzer	18	675	0	0.1	663	0.8	663	0.8	663	0.9	663	0.9
	Sokoban	16	2550	0	2	2534	65	2534	66	2534	86	2534	86
	Transport	13	2583	66	1	2493	62	2493	61	2493	294	2493	294
Jasper	Barman	8	1620	259	0.4	1220	5	1220	4	1220	72	1220	62
	Elevators	17	3777	58	2	3671	245	3671	240	3671	1156	3671	1138
	Floortile	5	261	16	0.01	237	0.3	237	0.3	237	1	237	1
	Nomystery	19	687	0	12	687	193	687	191	687	248	687	249
	Parking	20	2045	28	2	1983	2	1938	2	1938	4	1938	3
	Pegsol	20	644	0	0.06	644	0.6	644	0.8	644	0.8	644	0.8
	Scanalyzer	20	793	0	0.06	763	0.8	763	0.9	763	1	763	0.2
	Sokoban	16	4703	0	2	4477	103	4477	100	4477	328	4477	328
	Transport	7	1178	64	0.3	1069	20	1069	16	1069	110	1069	112

Table 6.2 Results of experiments on the plans for IPC14 domains found by the planners Yahsp3, Madagascar, Probe, Bfs-f, and Jasper. The column 'Plans' contains the number(Num) of the plans and the total length of them(T.Length). The column IAE contains the total number of the redundant inverse actions $\sum$ in the plans and the total run time in seconds $\Diamond$ required to find them. The following columns contains the total length of optimised plans Ω and the total time in seconds $\Diamond$ required for optimization process for the four evaluated algorithms

Planner	Domain	Plan		IAE		AE		AIAE		GAE		GAIAE	
		Num	T.Length	$\sum$	$\Diamond$	Ω	$\Diamond$	Ω	$\Diamond$	Ω	$\Diamond$	Ω	$\Diamond$
Yahsp3	Floortile	2	102	20	0	78	0.1	78	0.1	78	0.4	78	0.4
	Thoughtful	10	1556	4	0.1	1131	10	1131	12	1219	107	1219	101
Madagascar	Floortile	20	1861	160	0.3	1624	7	1624	5	1624	39	1624	36
	Thoughtful	5	210	4	0.06	177	0.5	177	0.5	177	0.7	177	0.7
	Barman	9	1790	26	0.7	1696	9	1696	8	1694	34	1694	33
Probe	Floortile	2	81	4	0	76	0.06	76	0.07	76	0.2	76	0.1
	Thoughtful	12	1283	0	0.78	1240	8	1240	7	1240	14	1240	13
	Transport	2	765	10	0.2	702	25	702	24	702	211	702	210
Bfs-f	Barman	20	3315	94	1	3207	12	3207	11	3205	46	3207	40
	Floortile	3	241	8	0.1	288	0.6	288	0.6	288	2	288	1
	Thoughtful	16	1746	0	2	1715	11	1715	11	1715	24	1715	21
	Transport	6	1558	42	0.6	1543	51	1543	48	1543	240	1543	242
Jasper	Floortile	2	90	0	0	82	0.08	82	0.07	82	0.3	82	0.3
	Thoughtful	17	1818	2	1	1753	9	1753	9	1753	22	21	
	Transport	6	1545	96	0.6	1397	48	1397	43	1397	462	1397	481

Table 6.3 Comparison of the total number of justified actions and the total number of redundant for plans of IPC2011 domains found by the planners Yahsp3, Madagascar, Probe, Bfs-f, and Jasper. The column 'Plans' contains the number(Num) of the tested plans and the total length of them(T.Length). The column JUA contains the total number of the justified unique actions in the tested plans, and the column Redundant contains the total number of the redundant actions in the tested plans.

Planner	Domain	Plans		JUA	Redundant
		Num	T.Length		
yahsp3	Barman	1	235	9	153
	Elevators	20	25985	718	14275
	Floortile	6	303	94	35
	Nomystery	9	257	149	2
	Parking	2	137	71	4
	Pegsol	20	603	281	0
	Scanalyzer	17	2278	210	380
	Transport	20	3432	670	375
m	Floortile	18	1663	485	211
	Nomystery	12	550	265	80
	Pegsol	17	479	222	0
	Scanlyzer	5	166	95	44
Probe	Barman	18	2780	215	48
	Elevators	17	4078	1069	120
	Floortile	4	181	62	9
	Nomystery	4	114	60	0
	Parking	11	1580	366	27
	Pegsol	19	626	277	0
	Scanalyzer	18	575	229	10
	Sokoban	16	5074	50	222
	Transport	13	3218	454	263
Bfs-f	Barman	20	3033	204	54
	Elevators	5	814	241	35
	Floortile	6	291	99	10
	Nomystery	16	610	341	0
	Parking	18	1528	731	0
	Pegsol	20	603	280	0
	Scanalyzer	18	675		12
	Sokoban	13	2550	48	16
	Transport	13	2583	516	90
Jasper	Barman	20	5081	93	300
	Elevators	17	3777	1257	106
	Floortile	5	261	79	24
	Nomystery	18	687	393	4
	Parking	20	2052	697	69
	Pegsol	20	644	284	0
	Scanalyzer	20	793	259	30
	Sokoban	16	4703	56	226
	Transport	7	1178	254	109

Table 6.4 Comparison of the total number of justified actions and the total number of redundant for plans of IPC2014 domains found by the planners Yahsp3, Madagascar, Probe, Bfs-f, and Jasper. The column 'Plans' contains the number(Num) of the tested plans and the total length of them(T.Length). The column JUA contains the total number of the justified unique actions in the tested plans, and the column Redundant contains the total number of the redundant actions $\sum$ in the tested plans.

Planner	Domain	Plan		Justified Unique	Redundant
		Num	T.Length		
Yahsp3	Floortile	2	102	29	24
	Thoughtful	10	1556	113	425
Madagascar	Floortile	20	1861	536	237
	Thoughtful	5	210	56	33
	Barman	9	1790	132	98
Probe	Floortile	2	81	27	5
	Thoughtful	12	1283	150	43
	Transport	2	765	115	63
Bfs-f	Barman	20	3315	284	108
	Floortile	4	241	77	13
	Thoughtful	16	1746	224	31
	Transport	6	1558	351	15
Jasper	Floortile	2	90	26	8
	Thoughtful	17	1818	214	65
	Transport	6	1545	327	148

GAIAE were adapted to work with this feature and implemented as U(AIAE), and U(GAIAE) respectively.

The detailed results of this evaluation are presented in Appendix A. New algorithms were compared with the original AE and GAE algorithms. Generally, the run times of UAIAE and UGAIAE compared with original ones AE and GAE respectively were improved in several cases when the number of justified unique actions was high especially in the Nomystery, Elevators, and Transport domains. From Table 6.5, it can observed that the run times were reduced up to 70% in the Nomystery domain.

6.6 Experiment 5: Plan Optimisation Technique(UAIAE) against Optimal Planners

This subsection analyses the results of a comparison between plan optimisation technique and optimal planners in some domains. Only the results of domains where optimal planners

Table 6.5 This table represents the percentage of reduced run time of UAIAE and UGAIA compared with original ones

Planner	UAIAE	UGAIAE
Yahsp3	71%	67%
Madagascar	68%	47%
Probe	62%	58%
Bfs-f	65%	74%
Jasper	70%	65%

were able to solve some of the problems are displayed. It compares between the quality of optimised plans by UAIAE (obtained by agile planners) and the quality of optimal plans (in term of the number of actions) to show how our method find optimal or sub optimal plans. In order to evaluate the efficiency of our method, it also compares between generation time of optimal plan against the total run time of generation and optimsation of plans. The results are shown in Tables 6.6, 6.7, 6.8, and 6.9.

Plan Quality

As can be seen from the Tables 6.6,6.7, 6.8, and 6.9, the length of the plans generated by agile planners and optimsed by UAIAE method is very close to length of the optimal ones in some cases and in others is same in the Floortile and Nomystery domains for plans generated by Madagascar , and the Scanalyzer domain for plans generated by Jasper. It was Only in the Sokoban domain that the length of the plans was not close.

Plan Generation and Optimisation Time

Generally, the generation and optimization times for satisficing plans were lees then from generation time of optimal plans. The most interesting results are those of the Madagascar planner in the Floortile domain and Jasper in the Scanalyzer Domain. All the plans take less than one second to be generated and optimised. On the oder hand, the generation and optimisation times of plans in the sokoban domain were more than generation time of optimal plans in some cases.

6.7 Experiment 6: Plan Optimisation Technique(UAIAE) via Anytime Planners

In this section, the performance of the fast AE family technique (UAIAE) with Lama (Richter and Westphal, 2010) and Mercury (Katz and Hoffmann, 2014) planners is evaluated. Since both planners perform multiple iterations of a heuristic search to improve their initial plans, the first plans generated by these planners were taken and optimised by UAIAE, and then

Table 6.6 This table Compares between optimal plans generated by SymbA*-2 and plans generated by Madagscar and optimised by UAIAE in the Floortile domain(IPC11). The column 'Optimal' contains the length of optimal plan Ω, and plan generation time $\Diamond$. The following column 'M+UAIAE' contains the length of the plan $\Omega1$ generated by Madagascar, the length of optimised $\Omega2$ plan by UAIAE, and the total time of generation and optimisation $\Diamond$.

Problem	Optimal		M+UAIAE		
	Ω	$\Diamond$	$\Omega1$	$\Omega2$	$\Diamond$
p01	35	0.1	39	35	0.02
p02	36	0.1	42	36	0.02
p03	44	0.1	51	44	0.03
p04	48	0.3	54	50	0.04
p05	45	0.8	50	49	0.2
p06	46	0.8	52	48	0.04
p07	60	2	75	60	0.1
p08	57	1	64	59	0.1
p09	74	5	91	76	0.1
p10	73	5	74	73	0.1
p11	75	15	85	77	0.1
p12	78	17	86	82	0.1

compared in terms of plan length and plan optimisation time with the best plans found by the planners. The best overall results for UAIAE were achieved in the Barman, Floortile and Transport domains. On the other hand, UAIAE was unable to find any redundant actions in the Pegsol and Nonmystery, and only found some of them in the Parking, Scanalyzer and Sokoban (see Appendix B for more details and results).

Table 6.10 shows a comparison of the plans in the Barman. As can be seen, Lama could only improve three solutions, but Mercury was able to improve all of its solutions. Lama is able to quickly find an initial solution, but fails to improve on it over time (Xie et al., 2013). However, UAIAE outperformed them both in terms of length and optimisation time. For the Elevators domain, Lama was not able to find any improvement on the initial solution, whereas Mercury was able to find some improvement. On the other hand, UAIAE was able to find some improvement in the Lama plans see Table 6.11 but unable to improve the Mercury plans. In the Floortile domain see Table 6.12, the plans optimised by UAIAE were very close in length to those optimised by Lama and Mercury. For the Transport domain, Lama was able to improve seven out of twenty plans, while Mercury was able to improve all twenty of them. Meanwhile, UAIE outperformed Lama in terms of both plan length and time in seventeen plans, but was unable to find any redundant actions in plans generated by Mercury.

Table 6.7 This table Compares between optimal plans generated by RIDA and plans generated by Madagscar and optimised by UAIAE in the Nomystery domain(IPC11). The column 'Optimal' contains the length of optimal plan Ω, and plan generation time ◊. The following column 'M+UAIAE' contains the length of the plan Ω1 generated by Madagascar, the length of optimised Ω2 plan by UAIAE, and the total time of generation and optimisation ◊.

Problem	Optimal		M+UAIAE		
	Ω	◊	Ω2	Ω2	◊
p01	18	60	23	20	2
p02	21	44	23	21	5
p03	25	67	65	28	82
p04	28	111	34	32	212
p05	34	94	45	37	8
p06	36	28	36	36	6
p08	40	45	47	44	161
p11	18	50	21	19	5
p12	21	38	21	21	3
p13	25	56	29	26	105
p14	28	86	28	28	4
p15	34	116	34	34	5
p16	36	27	40	36	4
p18	40	41	48	40	98
p20	48	64	56	48	132

Table 6.8 This table Compares between optimal plans generated by SymbA*-2 and plans generated by Jasper and optimised by UAIAE in the Scanalyzer domain(IPC11). The column 'Optimal' contains the length of optimal plan Ω, and plan generation time ◊. The following column 'J+UAIAE' contains the length of the plan Ω1 generated by Jasper, the length of optimised Ω2 plan by UAIAE, and the total time of generation and optimisation ◊.

Problem	Optimal		J+UAIAE		
	Ω	◊	Ω1	Ω2	◊
p01	10	11	14	10	1
p02	12	373	12	12	0.1
p03	20	2.6	30	30	0.03
p05	14	607	16	16	0.3
p06	14	6	24	16	1
p07	16	561	16	16	0.6
p08	18	5	22	22	0.8

Table 6.9 This table Compares between optimal plans generated by SMPS and plans generated by Jasper and optimised by UAIAE in the Sokoban domain(IPC11). The column 'Optimal' contains the length of optimal plan Ω, and plan generation time $\Diamond$. The following column 'P+UAIAE' contains the length of the plan $\Omega1$ generated by Probe, the length of optimised $\Omega2$ plan by UAIAE, and the total time of generation and optimisation $\Diamond$.

Problem	Optimal		P+UAIAE		
	Ω	$\Diamond$	$\Omega1$	$\Omega2$	$\Diamond$
1	161	33	205	193	2
2	151	1	242	228	2
3	185	36	245	245	2
4	289	17	309	309	9
5	190	10	324	324	5
6	299	126	367	345	16
7	429	1	429	429	38
8	207	13	335	323	7
9	354	547	592	562	94
10	147	147	273	273	109
11	83	1233	181	181	13
12	155	4	536	410	164
13	328	8	440	434	87
14	254	873	353	353	236

Table 6.10 Results of experiments on the plans for Barman domain(IPC11) found by Lama and and Mercury. The columns Lama and Mercury contains the first plan generated by them 'initial Solution'(the lenght of first plan $\Omega1$), and best plan generated by them 'Best Solution' (the length of optimized plan $\Omega2$), and First plan optimised by 'UAIA'(the length of optimized plan $\Omega3$).

Problem	Lama						Mercury					
	Initial Solution		Best Solution		UAIAE		Initial Solution		Best Solution		UAIAE	
	$\Omega1$	$\Diamond$	$\Omega2$	$\Diamond$	$\Omega3$	$\Diamond$	$\Omega1$	$\Diamond$	$\Omega2$	$\Diamond$	$\Omega3$	$\Diamond$
1	157	1	-	-	127	0.4	158	0.4	156	0.8	132	0.3
2	147	0.8	144	1491	124	0.3	138	0.5	-	-	122	0.2
3	177	1	167	706	143	0.5	155	0.7	-	-	123	0.3
4	154	0.8	-	-	136	0.3	142	0.5	141	1	128	0.2
5	162	1	-	-	136	0.4	159	0.8	157	1	135	0.3
6	222	4	-	-	156	0.7	163	1	162	2	145	0.3
7	146	0.4	-	-	132	0.4	157	0.8	-	-	137	0.3
8	160	1	-	-	150	0.4	156	1	154	3	138	0.3
9	188	2	-	-	157	0.6	202	2	200	5	162	0.5
10	165	1	-	-	157	0.6	183	1	172	3	161	0.5
11	231	191	-	-	173	0.8	191	2	189	5	159	0.5
12	208	2	-	-	165	0.8	196	1	193	3	162	0.5
13	252	9	-	-	197	1	213	1	209	3	181	0.7
14	231	4	-	-	181	1	200	1	198	70	170	0.7
15	213	10	-	-	199	1	288	5	275	9	215	1
16	225	2	-	-	187	1	232	1	231	3	232	0.9
17	273	7	-	-	208	2	248	2	-	-	198	1
18	216	1	-	-	190	1	222	0.9	-	-	185	0.9
19	191	2	-	-	172	0.8	165	3	-	-	159	0.5

Table 6.11 Results of experiments on the plans for Barman domain(IPC11) found by Lama. The column 'initial Solution' contains the length of the first plan Ω1 and plan generation time ◊ , and Column 'Best Solution' contains the length of optimized plan Ω2 and plan optimization time ◊. The following column contains the length of optimized plan Ω3 by UAIAE and optimization time ◊

Problem	Initial Solution		UAIAE	
	Ω	◊	Ω	◊
p01	80	0.1	80	0.4
p02	143	2	136	2
p03	156	2	151	2
p04	118	0.6	115	0.7
p05	116	0.4	115	0.8
p06	184	3	182	4
p07	173	2	161	3
p08	205	4	202	5
p09	198	5	196	4
p10	214	3	214	6
p11	254	18	241	9
p12	265	18	261	8
p13	284	64	274	10
p14	289	33	281	11
p15	288	26	286	11
p16	248	20	248	20
p17	312	45	300	31
p18	313	34	313	32
p19	380	155	369	45
p20	362	57	358	40

Table 6.12 Results of experiments on the plans for Floortile domain(IPC11) found by Mercury . The column 'initial Solution' contains the length of the first plan Ω1 and plan generation time ◊ , and Column 'Best Solution' contains the length of optimized plan Ω2 and plan optimization time ◊. The following column contains the length of optimized plan Ω3 by UAIAE and optimization time ◊

Problem	Initial Solution		Best Solution		UAIAE		Initial Solution		Best Solution		UAIAE	
	Ω	◊	Ω	◊	Ω	◊	Ω	◊	Ω	◊	Ω	◊
1	44	20	35	78	37	0.02	40	3	35	29	37	0.03
2	41	19	36	93	38	0.02	38	3	36	73	36	0.02
3	57	304	44	881	50	0.05	52	11	44	140	46	0.04
4	61	229	52	1407	52	0.06	56	444	-	-	52	0.05
5	51	429	45	720	47	0.05	53	125	45	653	53	0.05
6	52	424	52	158	**48**	0.05	56	44	46	467	56	0.06

Table 6.13 Results of experiments on the plans for Transport domain(IPC14) found by Lama. The column 'initial Solution' contains the length of the first plan Ω1 and plan generation time ◊ , and Column 'Best Solution' contains the length of optimized plan Ω2 and plan optimization time ◊. The following column contains the length of optimized plan Ω3 by UAIAE and optimization time ◊

Problem	Initial Solution		Best Solution		UAIAE	
	Ω	◊	Ω	◊	Ω	◊
1	99	2	-	-	96	1
2	120	2	-	-	113	1
3	166	7	163	109	166	3
4	150	5	-	-	139	2
5	205	6	-	-	191	3
6	255	66	233	76	222	5
7	260	235	-	-	245	7
8	163	4	-	-	144	3
9	325	154	-	-	306	15
10	202	11	-	-	187	5
11	173	7	-	-	165	3
12	191	7	-	-	174	4
13	186	9	-	-	175	4
14	569	610	481	685	529	73

6.8 Experiment 7: Plan Optimisation Technique versus Plan repair

In this experiment, the performance of the plan optimisation technique (UAIAE) was evaluated against a plan repair strategy using LPG (Gerevini et al., 2003). Since LPG restarts from a partial plan obtained from a valid one by removing some actions randomly and repairing invalidated plan, the evaluation was performed by taking valid plans generated by agile planners and invalidating them by randomly removing 25% of the actions to force the error, then inputting those invalidated plans to LPG in order to repair them. The lengths of the plans validated by LPG were compared with the lengths of the original ones optimised by UAIAE.

The figures 6.3, 6.4, 6.5, 6.6, 6.7 and 6.8, show the results obtained in IPC2011 domain problems solved by agile planners. The overall results show that the optimised plans were shorter than the repaired plans, except in the Sokoban domain, where the repaired plans were shorter than the original and optimised plans in some cases, and close to them in others. This applied to the total cost of each particular plan.

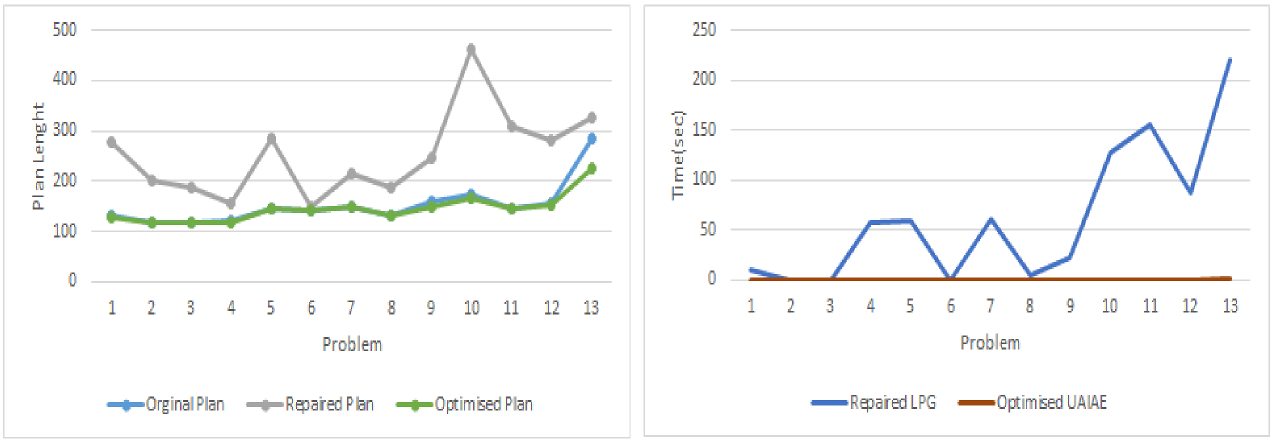


Fig. 6.3 Comparison between plan optimisation(UAIAE) and plan repair(LPG) in the Barman domain plans obtained by Jasper.On the left hand side, comparison of plan length and on the right hand side comparison of process time.

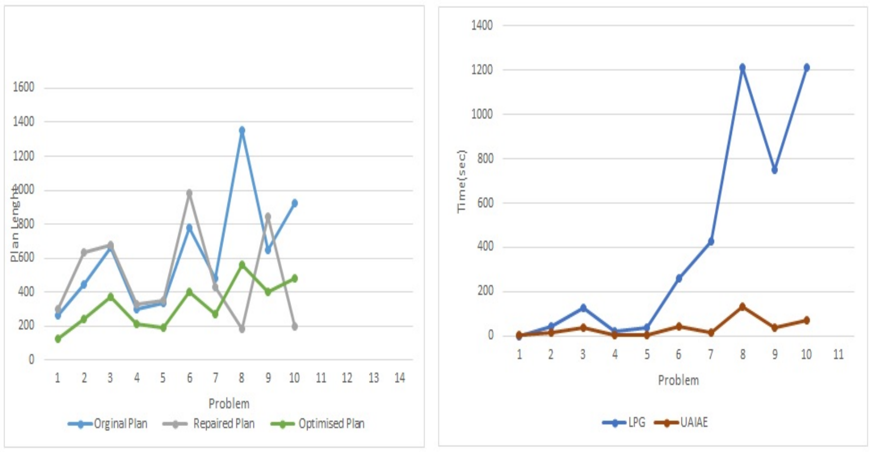


Fig. 6.4 Comparison between plan optimisation(UAIAE) and plan repair(LPG) in the Elevators domain plans obtained by Yahsp3. On the left hand side, comparison of plan length and on the right hand side comparison of process time.

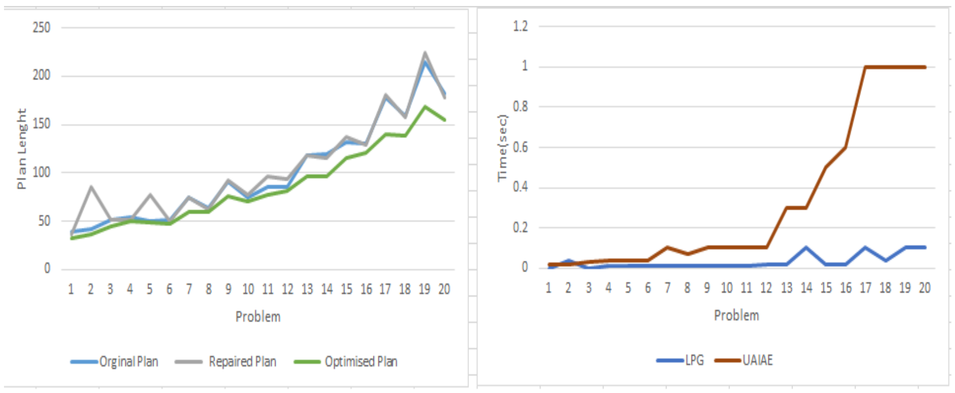


Fig. 6.5 Comparison between plan optimisation(UAIAE) and plan repair(LPG) in the Floortile domain plans obtained by Madagascar. On the left hand side, comparison of plan length and on the right hand side comparison of process time.

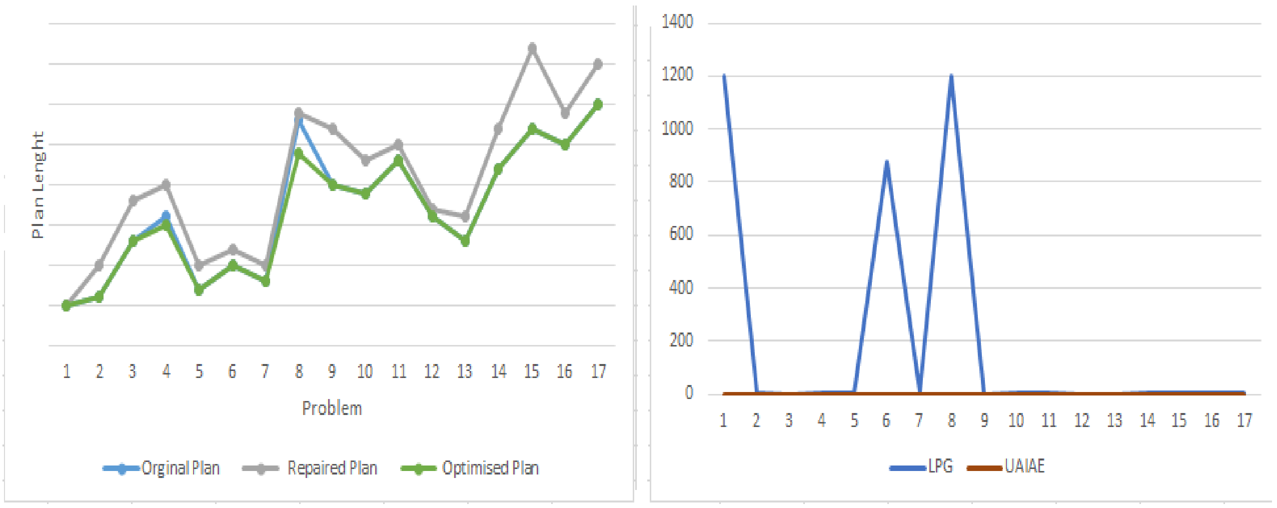


Fig. 6.6 Comparison between plan optimisation(UAIAE) and plan repair(LPG) in the Scanalyzer domain plans obtained by Probe. On the left hand side, comparison of plan length and on the right hand side comparison of process time.

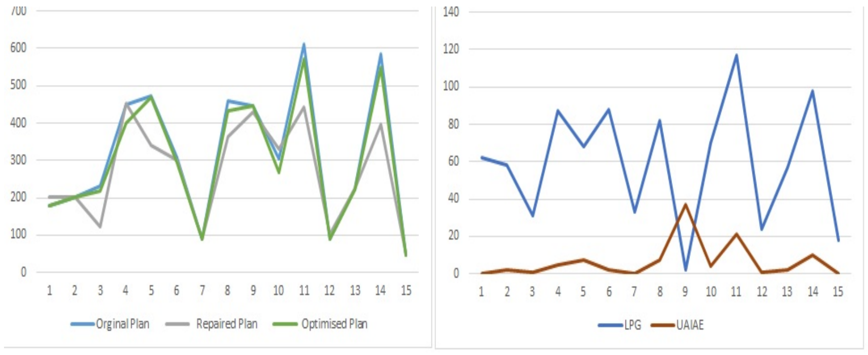


Fig. 6.7 Comparison between plan optimisation(UAIAE) and plan repair(LPG) in the Sokoban domain plans obtained by Jasper. On the left hand side, comparison of plan length and on the right hand side comparison of process time.

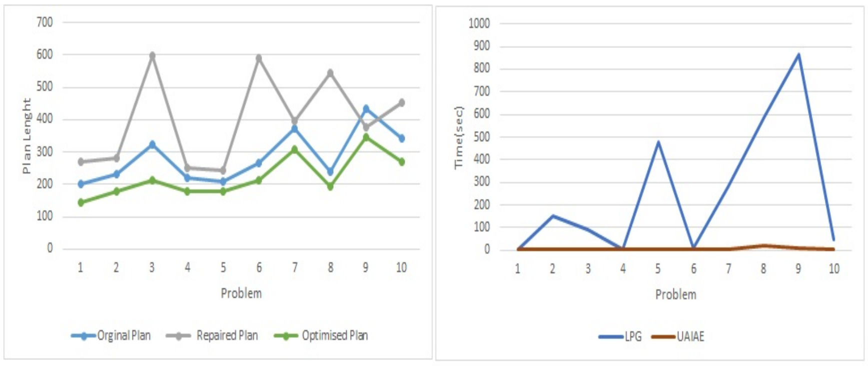


Fig. 6.8 Comparison between plan optimisation(UAIAE) and plan repair(LPG) in the Transport domain plans obtained by Yahsp3. On the left hand side, comparison of plan length and on the right hand side comparison of process time.

If the post-processing time is considered, it can be observed that the optimisation process is generally faster than the repair process. However, in some cases, the repair process was faster than the optimisation process, for example, in the Sokoban domain. An interesting observation was that when the number of justified unique actions in the plans was high, LPG was able to introduce plans that were close to both the original and optimised plans, for example, in the Floortile domain. On the other hand, when the number of justified unique actions was low, LPG introduced plans that were very different from the original and optimised plans, such as in the Barman domain. Therefore, it can be concluded that LPG plan repair works well when the number of JUA is high because of the plans that contains high number of JUA is high quality plans and original high quality plans led to repaired high quality plans. For more detailed results see Appendix C.

6.9 Discussion

The aim of extending action elimination and Greedy Action elimination was to reduce CPU-time of these algorithms while the keeping the same 'elimination power' (identifying and removing the same number of redundant actions). Clearly, the extended algorithms AIAE and GAIAE (considering situation where the inverse actions are redundant) remove the same number of redundant actions as AE and GAIAE respectively, and improve the run time in some cases where the number of inverse actions is high, such as the transport and Sokoban domains for plans generated by the Yahsp3 planner. In some other cases the improvement was not great, but was enough to to demonstrate the validity of our methods. On the other hand, in some cases the the time was not improved at all and in the worst cases was increased and added additional steps compared to the original algorithms. This happened for two reasons. The first reason was that, there were no redundant actions or a vary low number of redundant actions in the plans, such as,the Thoughtful domain plans generated by Madagascar. The Second reason may be that the inverse actions lie too far from each other in the plan. These algorithms will be more beneficial in specific cases in domains that have a high number of inverse actions where pairs of redundant inverse actions lie too close from each other in a plan.

Furthermore, adding a pre-step to plan optimisation process that identified the subset of justified unique (non-redundant)actions in a given plan was efficient in several cases, especially when the number of the justified unique actions was very high, such as Elevators, Nomyastery, and Transport domains. Clearly, adapting three approaches to work together in one algorithm (UAIAE) and four approaches in the other (UGAIAE) is guaranteed to provide optimised plans in less run time then AE, and GAE, but in term of plan quality their performance is same as the AE and GAE methods.

For comparison with optimal planners, UAIAE was able to introduce optimal or sub-optimal plans in less time in some cases. For compression with anytime planners, UAIAE outperforms these planners in several cases weather in term of quality or term of optimisation time. Even though , Lama and Mercury are given 30 minutes they find first solution and they were not able to improve it over the time. This time was identified by Xie et al. (2013) as the "unproductive time" of Lama in IPC-2011 domains(Barman and Elevators). On the other hand, in some cases UAIAE was not able to improve the plans.

However, the total run time was very high in some cases because the algorithms take input in PDDL format and this varies depending on the size of the plans and domains. The highest run time was in optimising plans from the Transport, Elevators, Nomyastery and Sokoban domains. On the other hand, the lowest run time was in optimising plans of Floortile and Barman.

Finally, it can be concluded that pre-optimisation techniques can be integrated with satisficing planners in order to find optimal or sub optimal plans in a polynomial time.

Chapter 7

Conclusion and Future Work

This chapter summarizes our contributions and proposes several directions to follow for future work.

7.1 Contributions

This thesis contribute to the topic of post planning plan optimisation as follow:

- It has provided an overview of the existing plan optimisation techniques and presented the current techniques that identify and remove redundant actions from a given plan in polynomial time, such as Action Elimination and Greedy Action Elimination. Such techniques are very important because of they serve as pre-process and a a complementary tool to more sophisticated plan optimisation techniques.

- It has extended Action Elimination and Greedy Action Elimination algorithms by incorporating Inverse Action Elimination feature into them and developed two new algorithms named AIAE and GAIAE respectively. This extension was accompanied by necessary theoretical foundations. New algorithms are beneficial to optimise plans in shorter time in some cases where the number of inverse actions is high such as transport domain. They will be more efficient when pairs of inverse actions are close to each other in the plan.

- It has presented the development of an algorithm to extract subset of the actions from a given plan that cannot be present at any set of redundant actions or cannot be removed from the plan. Again, this implementation has been accompanied with necessary theoretical foundations. The subset, which contains the actions that introduce unique facts to goal and the actions introduce unique facts to them named justified unique actions; this is an approach that, to the author's knowledge, this approach has never been previously explored.

- It has presented adding pre-process to the above extended methods (AIAE and GAIAE) and developing two new algorithm UAIAE and UGAIAE respectively. New algorithms firstly identify non-redundant actions (justified unique actions) in a given plan, then they identify the redundant actions. In several cases, this approach proved efficient in reducing CPU-time than the original methods.
- It has provided detailed experimental results that show pre-optimization process is very an important process to post planning optimisation, because it identifies and removes many redundant actions from a solution plan in polynomials time before applying more sophisticated plan optimisation technique.

7.2 Challenges and Future Work

Clearly, there has been significant progress during the last years in this area; however, post-planning plan optimization remains a challenge. In addition, although there are several algorithms that identify and remove redundant actions from a plan, determining a maximum set of redundant actions in a plan is intractable. In this section, we present some of challenges and directions to future work.

- **Optimal Planning with Macro-operators**

  A macro-operator is a sequence of actions that can be planned at one time like a single operator. In some instances, through the use of macro-operators, domains can be enhanced, with such operators encompassing a number of sequences of primitive operators. It also has been recognised that, during the course of the application of macro-operators throughout the search, plans may be generated at a much faster pace; however, this commonly goes hand-in-hand with the drawback of a loss in solution plans optimality. As has been noted in (Chrpa et al., 2012b), in some instances, there may be the opportunity to refine an optimal plan from (sub-optimal) plans through the adoption of macro-operators sourced from an optimal planning engine. In the Depots domain, macro-operators' lift-load and unload-drop' may be generated. Should there be an issue, such as the need to move a crate from one stack to another, the use of an optimal planning engine could facilitate a solution. Such a plan, i.e. (lift-load,unload-drop), can be unfolded to generate (lift,load,unload,drop) although this is not considered an optimal plan. It is apparent that load and unload are inverse actions, and thus can be established both as redundant and removed (Chrpa et al., 2012a). Nonetheless, should the depot comprise two or more hoists, such actions might not be inverse owing to the fact that the crate would be unloaded form the truck with a different hoist. In an instance such as this, the actions load and unload would not be disregarded from the plan owing to the fact that the precondition of the drop would not be satisfied. In order to resolve this, the argument referring to the drop action's hoist would be amended in line with the lift action.

A macro-operator refuel-fly can be generated in the Zeno domain, whereby, even in the case of utilising an optimal planning engine, it remains that an excessive number of refuel actions might be included in the unfolded solution plans. Comparably, deleting the plan's refuel action would mean the subsequent fly or zoom actions would be inapplicable; thus, their purposes in regards fuel levels would require modification. Accordingly, it would be suitable to direct attention towards classifying possible situations that might arise whilst planning with macro-operators using optimal planning engines, i.e. those without the presence of action costs. Devising or implementing existing optimisation approaches for such classes whilst ensuring their optimality is a notable challenge to undertake.

- **Characteristic Domains:**

  When considering the BlocksWorld domain, which is very well known, one plan non-optimality source is that of Sussman Anomaly (see Figure 7.1). As an example, if there are three blocks referred to as A, B and C, such that the initial state is on (C, A) and on Table(B) and the goal situation is on (A,B) and on (B,C), there would be two possibilities in regards to achieving goal atoms order, i.e. on (A,B) then on (B,C), or vice-versa. A number of different plans may be generated, namely i) hunstack(C,A), putdown(C), pickup(A), stack(A,B), unstack(A,B), putdown(A), pickup(B), stack(B,C), pickup(A), stack(A,B), and ii) hpickup(B), stack(B,C), unstack(B,C), putdown(B), unstack(C,A), putdown(C), pickup(A), stack(A,B), unstack(A,B), putdown(A), pickup(B), stack(B,C), pickup(A), stack(A,B)i. In such cases, an approach centred on establishing redundant actions can be adopted (Chrpa et al., 2012a); these are able to provide the most optimal of optimised plans. Such an approach is recognised as valuable in working in the ways the literature considers, specifically in regards more complicated instances, such as when, after stacking A to B, we might be moving blocks between different stacks and subsequently unstacking A from B. Notably, such an approach should be questioned in terms of whether it can guarantee optimal plans.

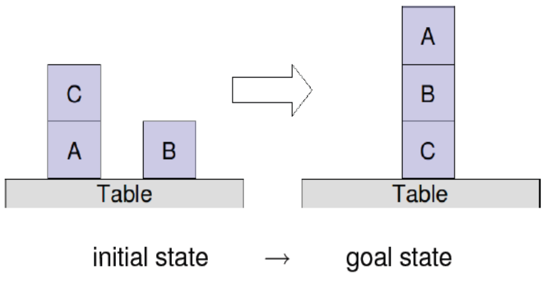


Fig. 7.1 An Example Blockword Task 2

When considering the Gold-miner domain, there is a need to progress through the maze in an effort to gather gold. In this case, there is the need to unblock cells, such as with the use of bomb or laser; notably, however, a bomb is the only way of unblocking a cell containing goal. Importantly, when unblocking using bomb, the bomb is consumed and, as a result, would need to be re-collected. Accordingly, the most valuable approach would be to select the laser for unblocking all essential cells, and then using the bomb only when there are gold-containing cells. Regardless of such suggestions, however, some planners might hold the preference of bomb utilisation as opposed to laser; this could result in non-optimal plans. In this case, there would need to be the use of an optimal plan, such as with the use of AIRS (Fikes and Nilsson, 1971); this can use an optimal (or almost optimal) planner to re-plan that particular element of the plan.

In regards the Zeno domain, as an example, we might utilise more refuel actions than is strictly required; nonetheless, as discussed earlier, disregarding essential refuel actions would mean the plan would be considered invalid. Thus, in order to make the plan valid, the fuel level arguments associated with zoom or fly actions would need to be modified. It is possible that an amended version of this technique could be devised in order to establish redundant actions (Chrpa et al., 2012b).

In the Depots domain, it might be decided that shifting a crate from one truck to another is not an optimal approach; rather than implementing two actions, namely unload and load, one additional drive action could be included. However, the unload and load actions cannot be simply replaced with the drive action as, in such an instance, the plan's validity might be lost. There is the need to propagate such a change as the crate will not be unloaded from the second truck, and the first truck would be located elsewhere. In this case, it might be more valuable to implement the PNGS (Nakhost and Müller, 2010) technique, although it might not work so well if such actions are positioned far from one another.

- **Learning Shortcut Rules:** Shortcut rules (Karpas and Domshlak, 2012) can be understood as a mapping between causal structures of planning operators (or actions) , such that structures are mapped to structures offering the benefit of lower costs, as established by the sum of action costs of operators or the number of operators. A particular shortcut rules case is that of plan rewrite rules (Nedunuri et al., 2011), which may be applied for replacing sub-plans (outcomes of actions in plans) by shorter (or less expensive) action outcomes. AIRS (Estrem and Krebsbach, 2012) is concerned with establishing possibly non-optimal sub-plans and replacing them with optimal (or almost optimal) ones. In this vein, pairs of weakly adjacent actions, i.e. those that may be adjacent in some change implemented in the plan, is established in (Chrpa et al., 2012b); these then may be replaced by a single action. In such an instance, replaceability may be explained as an action that might replace another action but that needs to have an equal or otherwise weaker precondition, equal or weaker negative effect, or equal or stronger positive effect. In some instances, however, this could be a very strong assumption to make as, in some instances, strongly preconditions could be satisfied. Accordingly, there is a need to differentiate between problem- or plan-specific shortcut rules or problem-independent rules. As is clear, shortcut rules may comprise far more complicated planning operator-related casual structures. However, the question centres on how shortcut rules may be learnt. One possibility is to learn when optimising plans online; nonetheless, this approach could have the drawback of consuming too much time owing to the fact there is a need to carry out a blind check on the significantly high number of casual structures in the operator. One further possibility is to establish potential non-optimal operator casual structures; this can be done through examining the schema of operators, which could provide the opportunity to establish different ways of achieving an atom or atoms. Nevertheless, there are various drawbacks that could hinder the learning of problem-specific shortcut rules: as an example, in the case of the Depots domain, a single drive action could replace two drive actions, with drive(t,a,c) replacing drive(t,a,b), drive(t,b,c). However, assuming an extensive of the domain, where there needs to be a road between locations so as to facilitate the truck moving, is applicable only when there is a road between a and c locations. Implementing shortcut rules could be tractable; nonetheless, it is debatable whether or not the extent of the polynomial could be excessive. Ensuring a pair of actions is replaced by a single one can take at worst O(l2) (where l is the number of actions placed in between the pair) (Chrpa et al., 2012b). Nonetheless, there may be a greater complexity recognised in complex shortcut rules. When taking into account situations where shortcut rules' actions are not aligned, replacing such actions could cause complicated as the new actions might be influenced by actions positioned between the old actions. As such, there is a need to examine how adopting shortcut rules, and the computational complexity of such, rises in line with size; in other words, involving more complex causal structures of operators). However, structures such as causal graph or relations such as mutexes might be useful in identifying suboptimal subsequences.

Developing, a new system for post-planning plan optimisation that combines new tools developed in this work with a more sophisticated plan optimisation technique that learns and applies shortcut rules would be an interesting direction for future work. The system could apply one of this work tools to identify and remove redundant actions, then apply the other method to learn and apply shortcut rules alternately.

- **Integration with Heuristic-based Planners:**

  Some of this work methods could be integrated with heuristic-based planners such as Lama. The method could be used as an intermediate step in the plan improvement process to identify and remove redundant actions from a solution plan that will be improved by the planner. Every time the planner tries to improve the plan, the method should be applied. This would help the planner to optimise the plan in a little time, as well as to improve the plans in some domains where the planner is unable to make any improvement.

- **Extension:**

  Even though the tools utilised in this thesis support only classical (STRIPS) planning,the intention is to extend this further in the further for non-classical planning, such as temporal planning . In this case, identification and removing of redundant actions will not change the plan‘total ordering. The Only problem would be that after removing redundant actions there is need to re-schedule the rest of the plan.

References

Balyo, T. (2014). The freelunch planning system entering ipc 2014. *The Eighth International Planning Competition (IPC-2014)*, pages 33–34.

Balyo, T., Bartak, R., and Surynek, P. (2012). Shortening plans by local re-planning. In *Tools with Artificial Intelligence (ICTAI), 2012 IEEE 24th International Conference on*, volume 1, pages 1022–1028. IEEE.

Balyo, T. and Chrpa, L. (2014). Eliminating all redundant actions from plans using sat and maxsat. In *ICAPS 2014 Workshop on Knowledge Engineering for Planning and Scheduling (KEPS)*.

Balyo, T., Chrpa, L., and Kilani, A. (2014). On different strategies for eliminating redundant actions from plans. In *Seventh Annual Symposium on Combinatorial Search*.

Bonet, B. and Geffner, H. (1999). Planning as heuristic search: New results. In *European Conference on Planning*, pages 360–372. Springer.

Bonet, B. and Geffner, H. (2001). Planning as heuristic search. *Artificial Intelligence*, 129(1):5–33.

Bylander, T. (1994). The computational complexity of propositional strips planning. *Artificial Intelligence*, 69(1-2):165–204.

Chrpa, L. (2010). Generation of macro-operators via investigation of action dependencies in plans. *The Knowledge Engineering Review*, 25(3):281–297.

Chrpa, L. and Barták, R. (2008). Towards getting domain knowledge: Plans analysis through investigation of actions dependencies. In *FLAIRS Conference*, pages 531–536.

Chrpa, L., McCluskey, T. L., and Osborne, H. (2012a). Determining redundant actions in sequential plans. In *Proceedings of ICTAI*, pages 484–491.

Chrpa, L., McCluskey, T. L., and Osborne, H. (2012b). Optimizing plans through analysis of action dependencies and independencies. In *ICAPS*.

Chrpa, L., Vallati, M., and McCluskey, T. L. (2014). MUM: A technique for maximising the utility of macro-operators by constrained generation and use. In *Proceedings of the Twenty-Fourth International Conference on Automated Planning and Scheduling, ICAPS 2014, Portsmouth, New Hampshire, USA, June 21-26, 2014*.

Culberson, J. C. and Schaeffer, J. (1998). Pattern databases. *Computational Intelligence*, 14(3):318–334.

Dean, T. L. and Wellman, M. P. (1991). *Planning and control*. Morgan Kaufmann Publishers Inc.

Edelkamp, S. and Kissmann, P. (2008). Gamer: Bridging planning and general game playing with symbolic search. *IPC6 booklet, ICAPS*.

Erol, K., Nau, D. S., and Subrahmanian, V. S. (1995). Complexity, decidability and undecidability results for domain-independent planning. *Artificial intelligence*, 76(1):75–88.

Estlin, T., Castano, R., Anderson, R., Gaines, D., Fisher, F., and Judd, M. (2003). Learning and planning for mars rover science.

Estrem, S. J. and Krebsbach, K. D. (2012). Airs: Anytime iterative refinement of a solution. In *FLAIRS Conference*.

Fikes, R. E. and Nilsson, N. J. (1971). Strips: A new approach to the application of theorem proving to problem solving. *Artificial intelligence*, 2(3-4):189–208.

Fink, E. and Yang, Q. (1992). Formalizing plan justifications.

Franco, S., Barley, M., and Riddle, P. (2014). Rida: In situ selection of heuristic subsets. *International Planning Competition (IPC)*, pages 93–96.

Gerevini, A., Saetti, A., and Serina, I. (2003). Planning through stochastic local search and temporal action graphs in lpg. *Journal of Artificial Intelligence Research*, 20:239–290.

Gerevini, A., Saetti, A., Serina, I., and Toninelli, P. (2004). Planning in pddl2. 2 domains with lpg-td. *International Planning Competition booklet (ICAPS-04)*.

Gerevini, A. and Serina, I. (2002). Lpg: A planner based on local search for planning graphs with action costs. In *AIPS*, volume 2, pages 281–290.

Ghallab, M., Nau, D., and Traverso, P. (2004). *Automated planning: theory & practice*. Elsevier.

Gupta, S. K., Bourne, D. A., Kim, K., and Krishnan, S. (1998). Automated process planning for sheet metal bending operations. *Journal of Manufacturing Systems*, 17(5):338.

Haslum, H. and Geffner, P. (2000). Admissible heuristics for optimal planning. In *Proceedings of the Artificial Intelligence Planning and Scheduling (AIPS)*, pages 140–149.

Helmert, M. (2003). Complexity results for standard benchmark domains in planning. *Artificial Intelligence*, 143(2):219–262.

Helmert, M. (2006a). The fast downward planning system. *J. Artif. Intell. Res.(JAIR)*, 26:191–246.

Helmert, M. (2006b). New complexity results for classical planning benchmarks. In *ICAPS*, pages 52–62.

Helmert, M. and Domshlak, C. (2009). Landmarks, critical paths and abstractions: what's the difference anyway? In *ICAPS*.

Helmert, M., Haslum, P., Hoffmann, J., and Nissim, R. (2014). Merge-and-shrink abstraction: A method for generating lower bounds in factored state spaces. *Journal of the ACM (JACM)*, 61(3):16.

Hoffmann, J. (2003). The metric-ff planning system: Translating“ignoring delete lists”to numeric state variables. *Journal of Artificial Intelligence Research*, 20:291–341.

Hoffmann, J. and Nebel, B. (2001). The ff planning system: Fast plan generation through heuristic search. *Journal of Artificial Intelligence Research*, 14:253–302.

Hoffmann, J., Porteous, J., and Sebastia, L. (2004). Ordered landmarks in planning. *Journal of Artificial Intelligence Research*, 22:215–278.

Karpas, E. and Domshlak, C. (2009). Cost-optimal planning with landmarks. In *IJCAI*, pages 1728–1733.

Karpas, E. and Domshlak, C. (2012). Optimal search with inadmissible heuristics. In *ICAPS*.

Katz, M. and Domshlak, C. (2010). Implicit abstraction heuristics. *Journal of Artificial Intelligence Research*, pages 51–126.

Katz, M. and Hoffmann, J. (2014). Mercury planner: Pushing the limits of partial delete relaxation. *Proceedings of the 8th International Planning Competition (IPC-2014).*

Katz, M., Hoffmann, J., and Domshlak, C. (2013). Who said we need to relax all variables? In *ICAPS*.

Kautz, H. A., Selman, B., et al. (1992). Planning as satisfiability. In *ECAI*, volume 92, pages 359–363.

Keyder, E. and Geffner, H. (2008). Heuristics for planning with action costs revisited. In *ECAI*, pages 588–592.

Keyder, E., Richter, S., and Helmert, M. (2010). Sound and complete landmarks for and/or graphs. In *ECAI*, pages 335–340.

Kilani, A. (2015). Improving the efficiency of plan optimisation techniques. In *Thirteenth Scandinavian Conference on Artificial Intelligence: SCAI 2015*, volume 278, page 182. IOS Press.

Kissmann, P., Edelkamp, S., and Hoffmann, J. (2014). Gamer and dynamic-gamer–symbolic search at ipc 2014. *Proceedings of the International Planning Competition (IPC)*, pages 77–84.

Kvarnström, J. and Doherty, P. (2000). Talplanner: A temporal logic based forward chaining planner. *Annals of mathematics and Artificial Intelligence*, 30(1):119–169.

Lipovetzky, N., Ramirez, M., Muise, C., and Geffner, H. (2014). Width and inference based planners: Siw, bfs (f), and probe. *Proceedings of the 8th International Planning Competition (IPC-2014).*

McDermott, D., Ghallab, M., Howe, A., Knoblock, C., Ram, A., Veloso, M., Weld, D., and Wilkins, D. (1998). Pddl-the planning domain definition language.

Nakhost, H. and Müller, M. (2010). Action elimination and plan neighborhood graph search: Two algorithms for plan improvement. In *ICAPS*, pages 121–128.

Nau, D. S. (2007). Current trends in automated planning. *AI magazine*, 28(4):43.

Nau, D. S., Au, T.-C., Ilghami, O., Kuter, U., Murdock, J. W., Wu, D., and Yaman, F. (2003). Shop2: An htn planning system. *Journal of artificial intelligence research*, 20:379–404.

Nedunuri, S., Cook, W. R., and Smith, D. R. (2011). Cost-based learning for planning. In *ICAPS 2011 Workshop on Planning and Learning*, pages 68–75.

Newell, A. and Simon, H. A. (1963). Computers and thought. *chap. GPS, a Program That Simulates Human Thought*, pages 279–293.

Nilsson, N. J. (1984). Shakey the robot. Technical report, SRI INTERNATIONAL MENLO PARK CA.

Porteous, J. and Cresswell, S. (2002). Extending landmarks analysis to reason about resources and repetition. In *Proceedings of the 21st Workshop of the UK Planning and Scheduling Special Interest Group (PLANSIG'02).*

Porteous, J., Sebastia, L., and Hoffmann, J. (2001). On the extraction, ordering, and usage of landmarks in planning. In *Proceedings of ECP-01.*

Richter, S., Helmert, M., and Westphal, M. (2008). Landmarks revisited. In *AAAI*, volume 8, pages 975–982.

Richter, S. and Westphal, M. (2010). The lama planner: Guiding cost-based anytime planning with landmarks. *Journal of Artificial Intelligence Research*, 39:127–177.

Rintanen, J. (2014). Madagascar: Scalable planning with sat. *Proceedings of the 8th International Planning Competition (IPC-2014).*

Siddiqui, F. H. and Haslum, P. (2015). Continuing plan quality optimisation. *Journal of Artificial Intelligence Research*, 54:369–435.

Siddiqui, F. H., Haslum, P., et al. (2013). Plan quality optimisation via block decomposition. In *IJCAI*, pages 2387–2393.

Smith, S. J., Nau, D., and Throop, T. (1998). Computer bridge: A big win for ai planning. *AI magazine*, 19(2):93.

Tate, A., Drabble, B., and Kirby, R. (1994). O-plan2: an open architecture for command, planning and control. In *Intelligent Scheduling*.

Torralba, A., Alcazar, V., Borrajo, D., Kissmann, P., and Edelkamp, S. (2014a). Symba: A symbolic bidirectional a planner. In *International Planning Competition*, pages 105–108.

Torralba, A., Alcázar, V., López, C. L., Borrajo, D., Kissmann, P., and Edelkamp, S. (2014b). Spm&s planner: Symbolic perimeter merge-and-shrink. *International Planning Competition (IPC)*, pages 101–104.

Vallati, M., Chrpa, L., Grzes, M., McCluskey, T. L., Roberts, M., and Sanner, S. (2015). The 2014 international planning competition: Progress and trends. *AI Magazine*, 36(3):90–98.

Vidal, V. (2014). Yahsp3 and yahsp3-mt in the 8th international planning competition. *Proceedings of the 8th International Planning Competition (IPC-2014)*, pages 64–65.

Westerberg, C. H. and Levine, J. (2001). Optimising plans using genetic programming. In *Proceedings of ECP*, pages 423–428.

Xie, F., Müller, M., and Holte, R. (2014). Jasper: the art of exploration in greedy best first search. *The Eighth International Planning Competition (IPC-2014)*, pages 39–42.

Xie, F., Valenzano, R. A., and Müller, M. (2013). Better time constrained search via randomization and postprocessing. In *ICAPS*.

Yang, Q. (2012). *Intelligent planning: a decomposition and abstraction based approach*. Springer Science & Business Media.

Appendix A

Agile Planners and Plan Optimisation Techniques

This appendix presents detailed results of experimental evaluation of the extended algorithms in this thesis using several benchmark domains, as well as several planning engines that participated in the Agile track of the International Planning Competition 2014

In this evaluation, six plan optimisation techniques were tested AE, AIAE, GAE, GAIAE, UAIAE, and UGAIAE. Furthermore IAE algorithm were used to identify redundant inverse actions and JUA algorithm was used to identify justified unique actions in a given plan. Different problems of IPC2011 and IPC2014 domains solved by different agile planners, and optimised by these methods. For each problem, tables contain plan length , plan generation time, number of justified unique actions and their extraction time, number of redundant inverse actions and their identifying time, the length of the optimised plan by AE, AIAE, GAE,GAIAE, UAIAE, and GAIAE, their optimiation time. Ω denote to plan length, $\Diamond$ denote to process time, and $\sum$ number of the actions. Some of tables contain dashes, which indicate to undefined results. The results were presented per problem for each domain and planner. Tow comparison terms were used:

- Comparison of the run time of AE and AIAE, and between GAE and GAIAE by taking into account the number of redundant inverse actions identified by IAE.

- Comparison of the run time of AE and UAIAE, and between GAE and GAIAE by taking into account the number of justified unique actions.

IPC11 Domains

Barman

For Barman problems the results are presented in Tables A.1, A.2, A.3 and A.4. As can be seen, the number of redundant inverse actions is high in plans obtained by Yahsp3, thus AIAE reduced the run times of AE were improved. With regard to justified unique actions, it was low in all plans obtained by all planners.

Table A.1 Results of experiments on the plan for Barman domain(IPC11) found by Yahsp3 planner

Problem	Plan		JUA		IAE		AE		AIAE	UAIAE	GAE		GAIAE	UGAIAE
	Ω	◊		◊	Ω	◊	Ω	◊	◊	◊	Ω	◊	◊	◊
1	235	84	9	0.05	153	0.07	137	0.6	0.4	0.4	137	20	16	15

Table A.2 Results of experiments on the plans for Barman domain(IPC11)) found by Probe planner

Problem	Plan		JUA		IAE		AE		AIAE	UAIAE	GAE		GAIAE	UGAIAE
	Ω	◊		◊	Ω	◊	Ω	◊	◊	◊	Ω	◊	◊	◊
1	130	0.3	9	0.03	2	0.04	128	0.2	0.2	0.2	128	0.5	0.5	0.4
2	117	0.1	10	0.03	0	0.03	117	0.2	0.2	0.1	117	0.2	0.2	0.1
3	118	0.1	9	0.03	0	0.03	118	0.2	0.2	0.1	118	0.2	0.2	0.2
4	120	0.2	9	0.02	2	0.03	118	0.2	0.2	0.1	118	0.4	0.4	0.3
5	144	0.1	12	0.03	0	0.04	144	0.3	0.3	0.3	144	0.3	0.3	0.3
6	143	0.1	11	0.03	0	0.04	143	0.3	0.3	0.2	143	0.3	0.3	0.2
7	150	0.1	11	0.03	0	0.04	150	0.3	0.3	0.3	150	0.3	0.4	0.3
8	132	0.2	12	0.03	2	0.03	130	0.2	0.2	0.2	130	0.5	0.5	0.4
9	159	1	12	0.04	2	0.05	149	0.4	0.4	0.4	149	2	2	1
10	172	0.6	13	0.04	6	0.05	166	0.5	0.5	0.4	166	2	1	1
11	145	0.2	12	0.03	0	0.04	145	0.3	0.3	0.3	145	0.3	0.3	0.4
12	155	0.3	12	0.04	4	0.05	151	0.4	0.4	0.3	151	1	1	1
13	170	0.6	14	0.04	2	0.05	168	0.5	0.5	0.4	168	1	1	0.9
14	200	0.5	12	0.05	2	0.06	198	0.7	0.7	0.6	198	1	1	1
15	187	2	13	0.05	4	0.06	181	0.7	0.6	0.5	181	2	2	2
16	177	1	14	0.05	6	0.05	171	0.6	0.6	0.5	171	2	2	2
17	197	0.8	14	0.06	2	0.07	195	0.7	0.7	0.6	195	1	1	1
18	164	0.8	16	0.05	4	0.05	160	0.5	0.5	0.4	160	1	1	1

Elevators

Results related to Elevators domain are shown in Tables A.5, A.6 and A.7.The number of inverse action was high in plans generated by Yahsp3 and run times were improved in several times. For justified unique actions, it was high in all plans generated by all planners, and run times reduction was clear especially in plans obtained by Yahsp3 and Probe.

Table A.3 Results of experiments on the plans for Barman domain(IPC11) found by BFS-F planner

Problem	Plan		JUA		IAE		AE		AIAE	UAIAE	GAE		GAIAE	UGAIAE
	Ω	◊		◊	Ω	◊	Ω	◊	◊	◊	Ω	◊	◊	◊
1	113	0.3	9	0.02	0	0.03	113	0.2	0.2	0.2	113	0.2	0.2	0.2
2	109	0.1	10	0.02	0	0.03	109	0.2	0.2	0.1	109	0.1	0.1	0.1
3	131	0.4	9	0.02	2	0.03	129	0.3	0.2	0.2	129	0.5	0.5	0.4
4	112	0.3	9	0.02	0	0.02	112	0.2	0.2	0.1	112	0.2	0.2	0.2
5	123	0.1	12	0.03	0	0.03	123	0.2	0.2	0.2	123	0.2	0.2	0.2
6	128	0.6	11	0.03	0	0.03	128	0.2	0.2	0.2	128	0.2	0.2	0.2
7	136	0.5	11	0.03	2	0.04	134	0.3	0.2	0.2	134	0.6	0.6	0.5
8	123	0.1	12	0.03	0	0.3	123	0.2	0.2	0.2	123	0.2	0.2	0.2
9	140	0.7	12	0.03	0	0.04	140	0.3	0.4	0.3	140	0.3	0.4	0.3
10	138	0.2	13	0.04	0	0.04	138	0.3	0.3	0.3	138	0.3	0.3	0.3
11	141	0.7	12	0.03	0	0.04	141	0.3	0.4	0.3	141	0.3	0.4	0.3
12	151	1	12	0.04	2	0.04	147	0.4	0.4	0.4	147	1	1	1
13	160	1	14	0.04	0	0.05	160	0.5	0.5	0.5	160	0.5	0.5	0.5
14	203	1	13	0.05	22	0.06	179	0.7	0.7	0.6	179	7	7	6
15	192	2	14	0.05	18	0.06	172	0.7	0.7	0.6	170	6	7	6
16	170	16	15	0.05	2	0.05	168	0.5	0.6	0.5	168	1	1	1
17	153	0.4	16	0.04	0	0.05	153	0.4	0.5	0.4	153	0.4	0.5	0.4

Table A.4 Results of experiments on the plans for Barman domain(IPC11) found by Jasper planner

Problem	Plan		JUA		IAE		AE		AIAE	UAIAE	GAE		GAIAE	UGAIAE
	Ω	◊		◊	Ω	◊	Ω	◊	◊	◊	Ω	◊	◊	◊
1	161	0.7	9	0.03	27	0.04	133	0.3	0.3	133	133	4	3	3
2	147	1	9	0.03	20	0.04	127	0.3	0.2	127	127	2	3	2
3	156	1	12	0.03	18	0.04	138	0.3	0.3	138	138	3	3	2
4	228	3	11	0.05	56	0.07	167	0.7	0.7	167	167	17	14	13
5	220	1	11	0.05	62	0.07	149	0.7	0.5	149	149	16	13	12
6	172	1	12	0.04	18	0.05	144	0.4	0.4	144	144	4	4	4
7	227	1	13	0.06	38	0.08	180	1	0.8	180	180	17	14	13
8	209	3	16	0.06	20	0.07	182	0.8	0.8	182	182	9	8	7

Floortile

As shown in tables A.8, A.9, A.10, A.11 and A.12, the number of redundant inverse actions was low therefor run times of AIAE and GAIAE did not improved compared with original AE and GAIAE. on the other hand, the number of justified unique actions was fair and there little improvement in run times.

Nomystery

For Nomystery domain, the results are shown in Tables A.13, A.14, A.15, A.16 and A.17. No redundant actions were found except plans obtained by Madagascar. As can shown in Table A.14, there is 36 redundant inverse actions in problem(p03) and it improved run time

Table A.5 Results of experiments on the plans for Elevators domain(IPC11) found by Probe planner

Problem	Plan		JUA		IAE		AE		AIAE	UAIAE	GAE		GAIAE	UGAIAE
	Ω	◊		◊	Ω	◊	Ω	◊	◊	◊	Ω	◊	◊	◊
1	87	0.3	29	0.03	0	0.03	87	0.6	0.6	0.4	87	0.6	0.6	0.4
2	187	4	31	0.1	6	0.1	171	5	5	0.4	171	41	40	32
3	173	5	54	0.1	2	0.1	169	4	4	3	169	20	20	14
4	116	1	51	0.06	2	0.04	113	1	1	0.6	113	4	3	2
5	138	1	37	0.06	4	0.05	132	2	1	1	132	5	4	3
6	216	11	57	0.1	0	0.1	215	7	7	5	215	17	17	12
7	163	6	46	0.1	2	0.09	160	4	4	3	160	13	13	10
8	271	22	64	0.2	2	0.1	259	12	12	9	259	77	77	58
9	269	24	74	0.2	0	0.1	268	12	12	9	268	26	26	19
10	317	78	86	0.3	6	0.2	303	18	18	13	303	135	136	100
11	319	98	78	0.3	0	0.2	316	19	19	13	316	83	83	58
12	391	121	91	0.4	12	0.3	359	27	27	21	359	273	338	265
13	344	154	105	0.4	2	0.2	339	22	22	16	339	122	122	87
14	348	147	91	0.4	0	0.2	345	23	23	16	345	100	100	67
15	318	113	80	0.6	2	0.4	313	39	39	29	313	214	215	159
16	421	268	95	1	0	0.5	409	72	72	55	409	462	464	351

Table A.6 Results of experiments on the plans for Elevators domain(IPC11) found by BFS-F

Problem	Plan		JUA		IAE		AE		AIAE	UAIAE	GAE		GAIAE	UGAIAE
	Ω	◊		◊	Ω	◊	Ω	◊	◊	◊	Ω	◊	◊	◊
1	89	0.6	29	0.04	0	0.02	87	0.7	0.7	0.5	87	1	1	1
2	117	1	53	0.07	0	0.04	117	1	1	0.8	117	1	1	0.8
3	141	2	45	0.07	6	0.05	135	2	1	1	135	6	6	4
4	189	14	48	0.1	18	0.1	171	6	6	4	171	60	52	38
5	278	77	66	0.2	6	0.1	269	14	14	10	269	71	66	51

of extended algorithm AIAE. On the other hand, the number of justified unique actions was very high in all plan obtained by all planners. It can be noticed that the run times of extended algorithms were reduced in all the cases.

Parking

The results are shown in Tables A.18, A.19, A.20 and A.21. As can be seen, no redundant actions were found in plans obtained by Bfs-f and there are some of them in others obtained by Yahsp3, Probe and Jasper. All the redundant actions that found are inverse expect one problem(12-048) found by probe(see Table A.19), and run times were not improved because no high number of redundant inverse actions. Although, the number of justified unique actions was very high the run times were not improved compared with the above results.

Table A.7 Results of experiments on the plans for Elevators domain(IPC11) found by Jasper

Problem	Plan		JUA		IAE		AE		AIAE	UAIAE	GAE		GAIAE	UGAIAE
	Ω	◊		◊	Ω	◊	Ω	◊	◊	◊	Ω	◊	◊	◊
1	79	0.1	29	0.03	0	0.02	79	0.5	0.5	0.3	79	0.5	0.5	0.4
2	142	0.9	47	0.1	2	0.07	138	3	2	2	138	6	6	4
3	161	1.6	57	0.1	2	0.08	159	4	4	3	159	9	9	6
4	116	0.4	54	0.07	2	0.03	114	1	1	0.6	114	2	2	1
5	112	0.4	44	0.06	0	0.03	111	1	1	0.6	111	2	2	1
6	186	2	63	0.2	4	0.09	182	5	5	3	182	16	16	11
7	151	1	49	0.1	2	0.07	149	4	3	2	149	8	7	5
8	197	2	66	0.2	4	0.1	192	6	5	4	192	26	25	17
9	226	3	73	0.2	10	0.1	216	8	8	5	216	53	52	35
10	218	3	79	0.3	4	0.1	211	7	7	5	211	32	31	20
11	259	10	89	0.3	2	0.1	256	11	11	7	256	38	38	24
12	281	15	90	0.4	4	0.1	275	13	13	9	275	57	57	39
13	293	7	103	0.5	4	0.1	289	15	15	10	289	51	50	32
14	267	21	85	0.7	6	0.2	261	26	26	19	261	115	114	81
15	339	37	96	1	2	0.3	327	42	42	31	327	312	305	223
16	396	70	112	1	10	0.4	359	50	49	35	359	321	318	212
17	354	52	121	1	0	0.4	353	48	48	33	353	108	106	71

Table A.8 Results of experiments on the plans for Floortile domain(IPC11) found by Yahsp3

Problem	Plan		JUA		IAE		AE		AIAE	UAIAE	GAE		GAIAE	UGAIAE
	Ω	◊		◊	Ω	◊	Ω	◊	◊	◊	Ω	◊	◊	◊
1	42	8	13	0.01	6	0.01	35	0.03	0.03	0.02	35	0.1	0.1	0.1
2	43	16	13	0.01	6	0.01	36	0.03	0.03	0.02	36	0.1	0.1	0.1
3	55	156	17	0.01	6	0.01	48	0.07	0.06	0.04	48	0.3	0.2	0.1
4	56	147	16	0.01	6	0.01	50	0.06	0.06	0.04	50	0.2	0.2	0.1
5	53	251	17	0.01	4	0.01	49	0.07	0.07	0.04	49	0.2	0.1	0.1
6	54	251	18	0.01	4	0.01	50	0.07	0.07	0.04	50	0.2	0.2	0.1

Pegsol

The results are presented in Tables A.22, A.23,A.24, A.25 and A.26. For all plans generated by all planners, no redundant actions were found , but the number of justified unique actions were high.

Scanalyzer

For Scanalyzer domain, the results are presented in Tables A.27, A.28, A.29, A.30, A.31 . It can be seen no inverse redundant actions in all plans obtained by all planners and there is few number of redundant actions in some plans generated by all planners expect plans obtained by probe. Looking at the number of justified unique action, an interesting result can be observed. In many cases, the number of justified unique actions are very close to plan length or all the actions in the plan justified unique actions especially in plans obtained by bfs and probe planners.

Table A.9 Results of experiments on the plans for Floortile domain(IPC11) found by Madagascar

Problem	Plan		JUA		IAE		AE		AIAE	UAIAE	GAE		GAIAE	UGAIAE
	Ω	◊		◊	Ω	◊	Ω	◊	◊	◊	Ω	◊	◊	◊
1	39	0	15	0	4	0	35	0.03	0.04	0.02	33	0.1	0.1	0.06
2	42	0	14	0	4	0	36	0.03	0.04	0.02	36	0.1	0.1	0.08
3	51	0	17	0	4	0	44	0.06	0.06	0.03	44	0.2	0.2	0.1
4	54	0	15	0.01	2	0	50	0.07	0.06	0.04	50	0.2	0.2	0.1
5	50	0.2	19	0.01	0	0	49	0.07	0.06	0.04	49	0.1	0.1	0.07
6	52	0	17	0.01	0	0	48	0.1	0.07	0.04	48	0.3	0.2	0.1
7	75	0.02	21	0.01	12	0.01	60	0.1	0.1	0.1	60	1	1	0.7
8	64	0.02	21	0.01	2	0.01	59	0.1	0.1	0.07	59	0.5	0.5	0.3
9	91	0.02	24	0.02	12	0.01	76	0.3	0.2	0.1	76	2	2	1
10	74	0	27	0.01	2	0.01	71	0.2	0.1	0.1	71	0.5	0.5	0.3
11	85	0.04	26	0.02	6	0.01	77	0.2	0.2	0.1	77	1	1	0.8
12	86	0.02	27	0.02	4	0.01	82	0.3	0.2	0.1	82	0.7	0.7	0.5
13	118	0.1	31	0.03	20	0.01	97	0.6	0.5	0.3	97	5	4	3
14	132	0.1	38	0.05	14	0.02	115	1	0.8	0.5	115	8	8	5
15	131	0.1	37	0.04	8	0.02	121	1	1	0.6	121	5	5	4
16	178	0.02	43	0.07	36	0.04	140	2	1	1	140	23	22	15
17	159	0.1	43	0.07	18	0.04	139	2	1	1	139	14	14	10
18	182	0.2	50	0.1	24	0.05	155	2	2	1	155	32	30	20

Table A.10 Results of experiments on the plans for Floortile domain(IPC11) found by Probe

Problem	Plan		JUA		IAE		AE		AIAE	UAIAE	GAE		GAIAE	UGAIAE
	Ω	◊		◊	Ω	◊	Ω	◊	◊	◊	Ω	◊	◊	◊
1	41	36	13	0	4	0	37	0.03	0.03	0.2	37	0.1	0.1	0.8
2	40	40	13	0	0	0	40	0.03	0.03	0.02	40	0.04	0.04	0.03
3	51	0	17	0.01	2	0	47	0.06	0.06	0.04	47	0.2	0.2	0.1
4	49	0	19	0.01	0	0	48	0.6	0.6	0.04	48	0.1	0.1	0.07

Sokoban

Tables A.32, A.33, A.34 shows results related to the Sokoban domain. As can be seen there are no inverse redundant actions in all plans obtained by all planners, and the number of justified unique actions is little and some case not exist. However, run times in both cases were not improved.

Transport

The results of transport domain are presented in Tables A.35, A.36, A.37, A.38. As can be seen the number of inverse redundant actions was very high in plan generated by Yahsp3, and run times were improved in some cases. For the number of justified unique actions, it was in all plan and run times were improved especially in plans generated by Yahsp3.

Table A.11 Results of experiments on the plans for Floortile domain(IPC11) found by BFS-F

Problem	Plan		JUA		IAE		AE		AIAE	UAIAE	GAE		GAIAE	UGAIAE
	Ω	◊		◊	Ω	◊	Ω	◊	◊	◊	Ω	◊	◊	◊
1	35	1	13	0	0	0	35	0.02	0.02	0.02	35	0.04	0.04	0.03
2	37	1	13	0	3	0	34	0.02	0.02	0.02	34	0.07	0.06	0.05
3	52	253	16	0	2	0	50	0.05	0.05	0.04	50	0.1	0.1	0.08
4	47	9	19	0.01	0	0	47	0.05	0.06	0.04	47	0.06	0.06	0.04
5	50	10	17	0.01	2	0	48	0.06	0.06	0.04	48	0.1	0.1	0.08
6	70	19	21	0.01	3	0	67	0.1	0.1	0.04	67	0.4	0.4	0.2

Table A.12 Results of experiments on the plans for Floortile domain(IPC11) found by Jasper

Problem	Plan		JUA		IAE		AE		AIAE	UAIAE	GAE		GAIAE	UGAIAE
	Ω	◊		◊	Ω	◊	Ω	◊	◊	◊	Ω	◊	◊	◊
1	45	3	12	0	2	0	43	0.03	0.03	0.03	43	0.1	0.1	0.08
2	47	5	12	0	4	0	42	0.03	0.03	0.02	42	0.1	0.1	0.1
3	53	274	16	0.01	4	0	46	0.06	0.06	0.04	46	0.2	0.2	0.1
4	54	108	17	0.01	4	0	47	0.07	0.07	0.05	47	0.3	0.3	0.2
5	62	82	22	0.01	2	0.01	59	0.1	0.1	0.07	59	0.3	0.3	0.2

Table A.13 Results of experiments on the plans for Nomystery domain(IPC11) found by Yahsp3

Problem	Plan		JUA	
	Ω	◊	Σ	◊
1	19	0.04	12	0.1
2	21	0.08	14	0.1
3	29	53	16	0.6
4	34	17	17	1
5	44	1.5	22	1
6	43	0.05	26	0.9
7	18	1	12	0.04
9	28	24	16	0.3

Table A.14 Results of experiments on the plans for Nomystery domain(IPC11) found by Madagascar

Problem	Plan		JUA		IAE		AE		AIAE	UAIAE	GAE		GAIAE	UGAIAE
	Ω	$\Diamond$	Σ	$\Diamond$	Σ	$\Diamond$	Ω	$\Diamond$	$\Diamond$	$\Diamond$	Ω	$\Diamond$	$\Diamond$	$\Diamond$
1	23	2	11	0.1	2	0.07	20	0.7	0.7	0.3	20	2	1	0.6
2	23	5	13	0.1	2	0.1	21	1	1	0.3	21	2	1	0.6
3	34	208	18	1	2	1	32	14	13	4	32	24	23	7
4	45	2	16	1	8	1	37	16	13	6	37	62	59	30
5	36	3	22	1	0	1	36	12	11	3	36	10	11	3
6	47	158	25	1	2	0.6	44	12	11	3	44	31	30	8
7	21	5	11	0.05	2	0.03	19	0.3	0.3	0.1	19	0.7	0.5	0.2
8	21	3	14	0.1	0	0.05	21	0.4	0.4	0.1	21	0.4	0.4	0.1
9	29	104	14	0.3	2	0.2	26	2	2	1	26	6	6	2
10	28	3	19	0.6	0	0.4	28	5	5	1	28	5	5	1
11	34	3	20	0.6	0	0.4	34	5	5	2	34	5	5	2
12	40	2	20	0.7	4	0.4	36	7	6	2	36	17	17	7

Table A.15 Results of experiments on the plans for Nomystery domain(IPC11) found by Probe

Problem	Plan		JUA	
	Ω	$\Diamond$	Ω	Σ
p01	21	0.02	12	0.1
p02	24	1	14	0.1
p03	34	179	16	0.6
p04	35	54	18	1

Table A.16 Results of experiments on the plans for Nomystery domain(IPC11) found by BfS-F

Problem	Plan		JUA	
	Ω	◊	∑	◊
p01	20	0.01	12	0.1
p02	22	0.02	14	0.1
p03	32	0.04	16	0.7
p04	34	5	18	1
p05	36	6	21	1
p06	40	3	22	1
p07	48	2	24	2
p08	53	3	26	1
p09	51	23	29	4
p10	57	19	30	2
p11	19	0.02	12	0.1
p12	22	0.7	14	0.1
p13	29	0.01	16	0.3
p14	32	3	19	0.7
p15	35	161	20	0.6
p16	38	31	22	0.7
p18	42	5	26	0.5

Table A.17 Results of experiments on the plans for Nomystery domain(IPC11) found by Jasper

Problem	Plan		JUA	
	Ω	◊	Σ	◊
p01	20	0.03	12	0.1
p02	22	0.05	14	0.1
p03	32	0.1	16	0.7
p04	33	0.1	18	1
p05	38	1	20	1
p06	45	32	19	1
p07	49	27	24	2
p08	42	24	27	1
p09	50	102	29	3
p10	53	0.3	30	2
p11	19	0.1	12	0.05
p12	22	0.5	14	0.1
p13	28	2	16	0.3
p14	30	0.1	19	0.6
p15	34	17	20	0.6
p16	37	16	22	0.7
p17	41	204	25	1
p18	42	102	26	0.5
p19	50	76	30	1

Table A.18 Results of experiments on the plans for parking domain(IPC11) found by Yahsp3

Problem	Plan		JUA		IAE		AE		AIAE	UAIAE	GAE		GAIAE	UGAIAE
	Ω	◊	Σ	◊	Σ	◊	Ω	◊	◊	◊	Ω	◊	◊	◊
1	62	131	28	0.03	2	0.03	62	0.02	0.03	0.02	60	0.1	0.1	0.07
2	75	70	43	0.03	2	0.05	73	0.1	0.05	0.03	73	0.1	0.1	0.07

Table A.19 Results of experiments on the plans for parking domain(IPC11) found by Probe

Problem	Plan		JUA		IAE		AE		AIAE	UAIAE	GAE		GAIAE	UGAIAE
	Ω	◊	Σ	◊	Σ	◊	Ω	◊	◊	◊	Ω	◊	◊	◊
1	130	68	26	0.05	4	0.04	126	0.1	0.1	0.1	126	0.3	0.3	0.3
2	153	32	19	0.06	2	0.06	151	0.2	0.2	0.2	151	0.4	0.3	0.3
3	156	286	23	0.05	0	0.06	156	0.2	0.2	0.2	156	0.2	0.2	0.2
4	173	242	24	0.06	0	0.07	163	0.2	0.2	0.2	163	0.5	0.4	0.4
5	159	172	27	0.05	2	0.06	157	0.2	0.2	0.2	157	0.4	0.4	0.3
6	149	95	33	0.05	0	0.05	149	0.1	0.1	0.1	149	0.1	0.2	0.1
7	93	103	42	0.02	0	0.02	93	0.08	0.08	0.06	93	0.1	0.1	0.06
8	152	92	38	0.05	2	0.05	150	0.2	0.2	0.1	150	0.4	0.3	0.3
9	122	28	35	0.04	0	0.04	122	0.1	0.1	0.1	122	0.1	0.1	0.1
10	165	222	47	0.06	0	0.06	165	0.2	0.2	0.2	165	0.2	0.3	0.1
11	128	156	52	0.04	4	0.04	121	0.1	0.1	0.1	121	0.5	0.5	2

Table A.20 Results of experiments on the plans for parking domain(IPC11) found by BFS-F

Problem	Plan		JUA	
	Ω	◊	Ω	◊
1	78	41	29	0.02
2	99	68	30	0.03
3	92	57	39	0.03
4	74	19	41	0.02
5	94	114	30	0.02
6	104	138	34	0.03
7	74	28	43	0.02
8	86	113	37	0.02
9	74	42	44	0.02
10	71	30	44	0.02
11	19	60	45	0.03
12	47	76	41	0.02
14	91	95	41	0.03
15	105	178	41	0.03
17	88	97	50	0.03
18	113	208	49	0.04

Table A.21 Results of experiments on the plans for parking domain(IPC11) found by Jasper

Problem	Plan		JUA		IAE		AE		AIAE	UAIAE	GAE		GAIAE	UGAIAE
	Ω	◊	Σ	◊	Σ	◊	Ω	◊	◊	◊	Ω	◊	◊	◊
1	130	14	22	0.05	0	0.1	130	0.1	0.1	0.1	130	0.1	0.1	0.1
2	60	3	29	0.01	0	0.03	60	0.03	0.03	0.02	60	0.03	0.03	0.03
3	63	7	35	0.01	0	0.04	63	0.04	0.04	0.03	63	0.04	0.04	0.03
4	124	22	24	0.04	2	0.1	122	0.1	0.1	0.1	122	0.2	0.2	0.2
5	69	6	37	0.02	0	0.04	69	0.04	0.04	0.03	69	0.04	0.04	0.03
6	68	7	37	0.01	0	0.04	68	0.04	0.05	0.03	68	0.04	0.05	0.03
7	90	10	34	0.02	0	0.07	90	0.07	0.08	0.05	90	0.07	0.08	0.06
8	75	11	37	0.02	0	0.05	75	0.05	0.06	0.04	75	0.05	0.06	0.04
9	88	8	32	0.02	0	0.07	88	0.06	0.08	0.05	88	0.07	0.08	0.06
10	79	10	32	0.02	2	0.06	77	0.06	0.07	0.05	77	0.1	0.1	0.1
11	141	47	34	0.04	8	0.1	133	0.1	0.2	0.1	133	1	0.8	0.6
12	92	24	34	0.02	2	0.1	90	0.08	0.1	0.06	90	0.1	0.1	0.1
13	84	17	40	0.02	2	0.07	82	0.07	0.08	0.05	82	0.1	0.1	0.1
14	111	33	37	0.03	2	0.1	109	0.1	0.1	0.1	109	0.2	0.2	0.1
15	142	60	43	0.05	0	0.2	142	0.2	0.2	0.1	142	0.2	0.2	0.1
16	108	28	34	0.03	2	0.1	106	0.1	0.1	0.1	106	0.2	0.1	0.1
17	113	32	38	0.03	0	0.1	113	0.1	0.1	0.1	113	0.1	0.1	0.1
18	148	26	43	0.03	0	0.1	107	0.1	0.1	0.08	107	0.1	0.1	0.1
19	146	102	36	0.05	4	0.2	142	0.2	0.2	0.1	142	0.6	0.6	0.4
20	121	122	39	0.04	4	0.1	117	0.1	0.1	0.1	117	0.4	0.4	0.3

Table A.22 Results of experiments on the plans for Pegsol domain(IPC11) found by Yahsp3

Problem	Plan		JUA	
	Ω	$\Diamond$	Ω	$\Diamond$
1	26	0.4	12	0.01
2	24	0.1	15	0.01
3	24	0.1	13	0.01
4	28	0.07	11	0.01
5	23	0.09	16	0.01
6	26	0.08	12	0.01
7	28	0.1	13	0.01
8	28	0.08	13	0.01
9	31	0.09	14	0.01
10	27	0.09	14	0.01
11	25	0.1	13	0.01
12	31	0.1	11	0.01
13	23	0.08	13	0.01
14	30	0.06	14	0.01
15	22	0.09	13	0.01
16	32	0.1	17	0.01
17	36	0.1	17	0.01
18	41	2	18	0.01
19	44	0.1	19	0.01
20	54	129	13	0.01

Table A.23 Results of experiments on the plans for Pegsol domain(IPC11) found by by Madagascar

Problem	Plan		JUA	
	Ω	◊	Ω	◊
1	26	68	12	0.01
2	23	37	16	0.01
3	29	30	13	0.01
4	26	135	14	0.01
5	25	0.8	15	0.01
6	25	6	12	0.01
7	29	22	13	0.01
8	27	70	13	0.01
9	33	15	12	0.01
10	27	4	11	0.01
11	28	10	12	0.01
12	28	155	11	0.01
13	23	29	12	0.01
14	31	63	14	0.01
15	24	4	14	0.01
16	37	61	14	0.01
17	38	23	14	0.01

Table A.24 Results of experiments on the plans for Pegsol domain(IPC11) found by Probe

Problem	Plan		JUA	
	Ω	$\Diamond$	Ω	$\Diamond$
1	27	0.5	11	0.1
2	28	0.3	11	0.1
3	27	0.09	14	0.1
4	28	0.2	11	0.1
5	24	0.02	15	0.1
6	26	0.02	13	0.1
7	29	0.02	14	0.1
8	27	0.02	13	0.1
9	29	0.02	13	0.1
10	28	0.01	12	0.1
11	29	0.09	12	0.1
12	32	0.4	11	0.1
13	25	0.04	12	0.1
14	31	0	14	0.1
15	24	0.03	13	0.1
16	32	0.04	14	0.1
17	37	0.04	22	0.1
18	39	0.4	20	0.1
19	45	0.03	17	0.1
20	59	1	15	0.1

Table A.25 Results of experiments on the plans for Pegsol domain(IPC11) found by BFS-F

Problem	Plan		JUA	
	Ω	◊	Ω	◊
1	28	0.4	11	0.01
2	24	0.4	15	0.01
3	22	0.3	16	0.01
4	27	0.6	11	0.01
5	25	0.01	15	0.01
6	22	0	10	0.01
7	27	0	14	0.01
8	26	0.04	13	0.01
9	31	0.1	18	0.01
10	26	0	10	0.01
11	26	0.01	14	0.01
12	29	0.05	11	0.01
13	24	0.06	13	0.01
14	31	0.2	12	0.01
15	25	0.1	13	0.01
16	34	0.07	12	0.01
17	33	0.3	15	0.01
18	41	0.3	18	0.01
19	43	0.9	23	0.01
20	59	1	16	0.01

Table A.26 Results of experiments on the plans for Pegsol domain(IPC11) found by Jasper

Problem	Plan		JUA	
	Ω	◊	Ω	◊
1	28	0.7	11	0.01
2	29	0.06	11	0.01
3	28	0.5	13	0.01
4	27	0	11	0.01
5	25	0	15	0.01
6	28	0.1	13	0.01
7	28	0	14	0.01
8	27	0.02	13	0.01
9	32	0	14	0.01
10	29	0	8	0.01
11	29	0.02	14	0.01
12	31	0.04	15	0.01
13	24	0.01	13	0.01
14	34	0.08	14	0.01
15	25	0.1	13	0.01
16	35	0.02	19	0.01
17	33	0.8	23	0.01
18	44	0.01	18	0.01
19	29	0.02	15	0.01
20	59	0.06	17	0.01

Table A.27 Results of experiments on the plans for Scanalyzer domain(IPC11) found by Yahsp3

Problem	Plan		JUA		IAE		AE		AIAE	UAIAE	GAE		GAIAE	UGAIAE
	Ω	◊	Σ	◊	Σ	◊	Ω	◊	◊	◊	Ω	◊	◊	◊
1	96	10	5	0.03	0	0.02	80	0.05	0.05	0.04	80	0.1	0.1	0.1
2	94	0.1	11	0.03	0	0.02	74	0.1	0.1	0.1	74	1	1	1
3	56	0.08	8	0.01	0	0	44	0.03	0.03	0.02	44	0.1	0.1	0.1
4	92	0.09	10	0.02	0	0.01	66	0.07	0.08	0.06	66	0.3	0.3	0.3
5	124	0.2	14	0.04	0	0.03	86	0.3	0.3	0.2	86	3	3	2
6	96	6	7	0.02	0	0.01	80	0.04	0.05	0.04	80	0.1	0.1	0.1
7	160	0.4	16	0.05	0	0.04	128	0.6	0.6	0.5	128	7	6	5
8	130	4	8	0.02	0	0.02	118	0.08	0.09	0.08	118	0.3	0.3	0.3
9	146	0.08	12	0.03	0	0.02	122	0.2	0.2	0.1	122	2	1	1
10	124	0.1	14	0.02	0	0.02	86	0.2	0.2	0.1	86	2	2	1
11	158	0.1	15	0.03	0	0.03	144	0.3	0.3	0.2	144	3	2	2
12	76	0.09	13	0.01	0	0.01	66	0.08	0.09	0.07	66	0.4	0.4	0.3
13	60	0.08	9	0.01	0	0.01	44	0.05	0.05	0.04	44	0.2	0.2	0.2
14	164	0.2	17	0.03	0	0.03	128	0.5	0.5	0.4	128	7	6	5
15	238	0.2	16	0.06	0	0.06	216	0.9	0.8	0.7	216	9	8	8
16	156	4	18	0.04	0	0.04	136	0.6	0.6	0.5	136	5	5	4
17	308	0.3	17	0.09	0	0.1	280	2	1	1	280	21	21	20

Table A.28 Results of experiments on the plans for Scanalyzer domain(IPC11) found by Madagascar

Problem	Plan		JUA		IAE		AE		AIAE	UAIAE	GAE		GAIAE	UGAIAE
	Ω	◊	Σ	◊	Σ	◊	Ω	◊	◊	◊	Ω	◊	◊	◊
1	10	5	6	0	0	0	10	0	0	0	10	0	0	0
2	22	0.3	12	0	0	0	18	0.04	0.04	0.01	18	0.05	0.08	0.02
3	40	0.02	8	0	0	0	38	0.02	0.02	0.01	38	0.02	0.04	0.02
4	66	11	13	0.01	0	0.01	48	0.06	0.06	0.03	48	0.1	0.2	0.1
5	28	0.8	14	0.01	0	0	28	0.06	0.06	0.02	28	0.04	0.05	0.02

Table A.29 Results of experiments on the plans for Scanalyzer domain(IPC11) found by Probe

Problem	Plan		JUA		IAE		AE		AIAE	UAIAE	GAE		GAIAE	UGAIAE
	Ω	◊	Σ	◊	Σ	◊	Ω	◊	◊	◊	Ω	◊	◊	◊
1	10	1	8	0	0	0	10	0	0	0	10	0	0.01	0
2	12	0.1	12	0	0	0	12	0.02	0.02	0.01	12	0.02	0.02	0.01
3	26	0.01	10	0	0	0	26	0.01	0.01	0.01	26	0.01	0.01	0.01
4	32	0.04	12	0.01	0	0	30	0.02	0.02	0.02	30	0.04	0.04	0.03
5	14	0.5	14	0.01	0	0	14	0.02	0.02	0.01	14	0.02	0.02	0.01
6	20	1	11	0	0	0	20	0.01	0.01	0.01	20	0.01	0.01	0.01
7	16	1	12	0.01	0	0	16	0.03	0.03	0.01	16	0.03	0.04	0.01
8	56	6	2	0.01	0	0.01	48	0.02	0.02	0.03	48	0.05	0.06	0.06
9	40	0.1	16	0.01	0	0.01	40	0.03	0.03	0.02	40	0.02	0.03	0.02
10	38	0.2	13	0.01	0	0.01	38	0.04	0.04	0.03	38	0.04	0.04	0.02
11	46	0.3	19	0.01	0	0.01	46	0.04	0.04	0.03	46	0.04	0.04	0.02
12	32	0.1	11	0.01	0	0.01	32	0.02	0.02	0.01	32	0.02	0.02	0.03
13	26	0.04	9	0.01	0	0	26	0.01	0.01	0.01	26	0.01	0.01	0.01
14	44	0.8	15	0	0	0	44	0.06	0.06	0.04	44	0.06	0.06	0.01
15	54	1	22	0.01	0	0.01	54	0.07	0.07	0.05	54	0.07	0.07	0.04
16	50	2	17	0.01	0	0.01	50	0.1	0.1	0.06	50	0.1	0.1	0.05
17	60	2	26	0.02	0	0.01	60	0.1	0.1	0.08	60	0.1	0.1	0.06

Table A.30 Results of experiments on the plans for Scanalyzer domain(IPC11) found by BFS-F

Problem	Plan		JUA	
	Ω	◊	Σ	◊
1	10	1	6	0
2	12	0.3	12	0
3	32	0	13	0
4	42	0.1	18	0.01
5	14	2	14	0.01
6	34	29	5	0.01
7	16	6	16	0.01
8	42	4	4	0.01
9	52	0.6	21	0.01
10	38	0.4	13	0.01
11	62	2	26	0.01
12	32	0.1	11	0
13	26	0.1	9	0
14	44	1	15	0.01
15	72	10	29	0.02
16	50	4	17	0.01
17	82	39	34	0.02
18	15	47	14	0

Table A.31 Results of experiments on the plans for Scanalyzer domain(IPC11) found by Jasper

Problem	Plan		JUA		IAE		AE		AIAE	UAIAE	GAE		GAIAE	UGAIAE
	Ω	◊	Σ	◊	Σ	◊	Ω	◊	◊	◊	Ω	◊	◊	◊
1	14	1	7	0	0	0	10	0	0	0	10	0.01	0.01	0.01
2	12	0.1	12	0	0	0	12	0.01	0.01	0	12	0.02	0.01	0.01
3	30	0.02	12	0.01	0	0	30	0.01	0.01	0.01	30	0.01	0.01	0.02
4	42	0.06	13	0.01	0	0	42	0.02	0.02	0.02	42	0.04	0.04	0.04
5	16	0.3	16	0.01	0	0	16	0.02	0.02	0.01	16	0.03	0.03	0.01
6	24	1	3	0.01	0	0	16	0	0	0	16	0.02	0.01	0.02
7	16	0.6	16	0.01	0	0	16	0.02	0.03	0.01	16	0.03	0.03	0.01
8	22	0.8	6	0.01	0	0	22	0	0	0.01	22	0.01	0.01	0.02
9	50	0.1	16	0.01	0	0	50	0.04	0.04	0.03	50	0.05	0.05	0.05
10	40	0.2	13	0.01	0	0	38	0.05	0.05	0.03	38	0.1	0.1	0.1
11	60	0.3	19	0.01	0	0.01	60	0.07	0.07	0.05	60	0.1	0.08	0.06
12	34	0.1	11	0.01	0	0	32	0.03	0.03	0.02	32	0.06	0.06	0.04
13	26	0.04	9	0.01	0	0	26	0.01	0.01	0.01	26	0.02	0.02	0.01
14	46	0.4	15	0.01	0	0.01	44	0.08	0.08	0.05	44	0.2	0.1	0.1
15	72	0.8	22	0.01	0	0.01	72	0.1	0.1	0.09	72	0.1	0.1	0.1
16	50	1	17	0.01	0	0.01	50	0.1	0.1	0.06	50	0.1	0.1	0.1
17	82	1	25	0.01	0	0.01	82	0.2	0.2	0.1	82	0.2	0.2	0.1
18	39	38	7	0.01	0	0	31	0.03	0.03	0.03	31	0.1	0.1	0.1
19	55	29	8	0.01	0	0.01	55	0.04	0.04	0.03	55	0.04	0.04	0.04
20	63	25	12	0.01	0	0.01	59	0.04	0.05	0.04	59	0.1	0.1	0.1

Table A.32 Results of experiments on the plans for Sokoban domain(IPC11) found by Probe

Problem	Plans		JUA		IAE		AE		AIAE	UAIAE	GAE		GAIAE	UGAIAE
	Ω	◊	Σ	◊	Σ	◊	Ω	◊	◊	◊	Ω	◊	◊	◊
Problem	Plan		JUA		IAE		AE		AIAE	UAIAE	GAE		GAIAE	UGAIAE
1	205	1	4	0.1	0	0.01	193	1	1	1	193	3	3	3
2	242	1	0	0.1	0	0.1	228	2	2	2	228	5	5	5
3	132	0.4	7	0.1	0	0.1	132	0.8	0.8	0.7	132	0.8	0.8	0.7
4	245	3	2	0.1	0	0.1	245	2	2	2	245	2	2	2
5	309	6	1	0.1	0	0.2	309	3	3	3	309	3	3	3
6	324	2	1	0.1	0	0.04	324	3	3	3	324	3	3	3
7	111	1	8	0.1	0	0.2	111	0.3	0.3	0.3	111	0.3	0.3	0.3
8	367	10	3	0.04	0	0.6	345	6	6	6	345	18	18	18
9	429	0.6	2	0.1	0	0.2	429	37	37	37	429	37	37	37
10	335	1	2	0.5	0	0.6	323	6	6	6	323	18	18	18
11	592	68	4	0.2	0	0.1	562	26	26	26	562	82	82	82
12	273	106	6	0.5	0	0.1	273	3	3	3	273	3	3	3
13	181	11	2	0.1	0	0.4	181	2	2	2	181	2	2	2
14	536	156	2	0.3	0	0.3	410	8	8	8	408	40	40	40
15	440	79	2	0.2	0	0.2	434	6	6	6	434	13	13	13
16	353	230	4	0.1	0	0.1	353	6	6	6	353	6	6	6

Table A.33 Results of experiments on the plans for Sokoban domain(IPC11) found by BFS-F

Problem	Plan		JUA
	Ω	◊	Σ
1	185	25	4
2	215	5	0
3	112	2	7
4	203	7	0
5	285	222	0
6	262	12	2
7	94	0.2	9
9	429	0.6	2
11	496	18	2
13	79	20	4
14	137	3	2
15	53	0.5	16

Table A.34 Results of experiments on the plans for Sokoban domain(IPC11) found by Jasper

Problem	Plan		JUA		IAE		AE		AIAE	UAIAE	GAE		GAIAE	UGAIAE
	Ω	◊	Σ	◊	Σ	◊	Ω	◊	◊	◊	Ω	◊	◊	◊
1	177	5	4	0.1	0	0.07	177	1	1	0.1	177	1	0.7	0.7
2	201	1	0	0.1	0	0.1	201	2	2	2	201	2	2	2
3	232	0.5	7	0.1	0	0.1	218	1	1	1	218	5	5	5
4	451	6	0	0.2	0	0.2	401	5	5	5	401	25	25	25
5	473	20	0	0.2	0	0.2	469	6	6	7	469	13	13	13
6	308	3	2	0.1	0	0.1	294	3	3	2	294	9	9	9
7	90	0.3	10	0.03	0	0.2	90	0.2	0.2	0.2	90	0.2	0.2	0.2
8	460	82	3	0.2	0	0.5	434	7	7	7	434	26	26	26
9	445	0.1	2	0.6	0	0.1	445	38	37	37	445	41	41	41
10	305	4	2	0.2	0	0.5	267	4	4	4	267	26	26	25
11	612	11	4	0.6	0	0.04	572	22	22	21	572	134	134	134
13	91	29	4	0.04	0	0.1	91	0.6	0.5	0.5	91	0.6	0.6	0.6
14	225	47	2	0.1	0	0.03	221	3	2	2	221	6	6	5
15	586	138	2	0.4	0	0.4	550	10	10	10	550	39	40	39
16	47	35	14	0.03	0	0.02	47	0.2	0.2	0.1	47	0.2	0.2	0.1

Table A.35 Results of experiments on the plans for Transport domain(IPC11) found by Yahsp3

Problem	Plan		JUA		IAE		AE		AIAE	UAIAE	GAE		GAIAE	UGAIAE
	Ω	◊	∑	◊	∑	◊	Ω	◊	◊	◊	Ω	◊	◊	◊
1	201	0.2	27	0.1		0.06	143	4	3	2	143	58	51	44
2	230	0.2	28	0.1		0.07	177	4	3	3	177	68	62	54
3	322	0.4	31	0.1		0.1	214	9	9	7	212	95	95	85
4	222	0.1	21	0.07		0.06	179	4	3	2	179	34	35	32
5	207	0.1	26	0.06		0.06	177	3	3	2	177	22	31	28
6	265	0.2	31	0.1		0.09	211	6	5	4	213	77	95	87
7	371	0.4	29	0.1		0.1	309	15	14	11	309	244	255	234
8	241	0.3	36	0.1		0.09	193	5	5	4	199	25	31	27
9	435	0.7	31	0.2		0.2	348	23	22	19	348	411	331	371
10	343	0.5	40	0.1		0.1	269	12	11	8	293	83	209	184
11	298	0.4	37	0.1		0.1	225	8	7	5	225	161	150	130
12	297	0.4	38	0.1		0.1	248	9	8	6	259	47	95	80

Table A.36 Results of experiments on the plans for Transport domain(IPC11) found by Probe planner

Problem	Plan		JUA		IAE		AE		AIAE	UAIAE	GAE		GAIAE	UGAIAE
	Ω	◊	∑	◊	∑	◊	Ω	◊	◊	◊	Ω	◊	◊	◊
1	143	4	33	0.1	2	0.1	137	2	2	1	137	5	5	4
2	171	8	39	0.1	4	0.1	155	3	3	2	155	13	13	9
3	221	20	30	0.1	10	0.1	180	6	5	4	180	28	34	27
4	182	3	19	0.1	4	0.1	175	3	3	2	175	9	8	7
5	265	7	24	0.1	6	0.1	239	5	5	4	239	31	30	27
6	298	21	27	0.1	10	0.1	270	9	8	7	270	48	47	43
7	360	36	37	0.1	2	0.1	354	14	14	13	354	28	28	25
8	193	15	39	0.1	4	0.1	180	4	4	3	180	20	19	15
9	388	48	34	0.1	6	0.1	341	21	20	18	341	200	197	179
10	270	31	43	0.1	4	0.1	249	8	7	6	243	44	43	34
11	238	20	45	0.1	6	0.1	219	6	5	4	219	21	20	16
12	234	26	42	0.1	2	0.1	221	6	5	4	221	28	28	22
13	255	21	42	0.1	8	0.1	235	7	6	5	235	32	31	24

Table A.37 Results of experiments on the plans for Transport domain(IPC11) found by BFS-F

Problem	Plan		JUA		IAE		AE		AIAE	UAIAE	GAE		GAIAE	UGAIAE
	Ω	◊	∑	◊	∑	◊	Ω	◊	◊	◊	Ω	◊	◊	◊
1	127	5	35	0.1	2	0.04	124	2	1	1	124	4	4	3
2	133	12	38	0.1	2	0.04	128	2	2	1	128	5	5	3
3	157	20	41	0.1	4	0.04	151	3	3	2	151	10	9	6
4	113	3	30	0.1	4	0.03	108	1	1	0.8	108	4	3	2
5	197	4	30	0.04	4	0.04	193	3	3	2	193	10	9	7
6	230	19	36	0.1	8	0.1	218	5	5	4	218	35	35	28
7	301	34	40	0.1	4	0.1	293	10	12	9	293	41	40	35
8	165	26	46	0.1	4	0.1	161	3	3	2	161	10	9	6
9	299	45	40	0.1	8	0.1	288	13	12	10	288	72	81	68
10	236	51	42	0.1	10	0.1	225	7	6	5	225	46	45	36
11	202	31	48	0.1	4	0.1	198	4	4	3	198	13	12	9
12	176	23	45	0.1	0	0.1	175	3	3	2	175	7	6	4
13	247	38	45	0.1	12	0.1	231	6	6	5	231	37	36	29

Table A.38 Results of experiments on the plans for Transport domain(IPC11) found by Jasper

Problem	Plan		JUA		IAE		AE		AIAE	UAIAE	GAE		GAIAE	UGAIAE
	Ω	◊	Σ	◊	Σ	◊	Ω	◊	◊	◊	Ω	◊	◊	◊
1	117	1	32	0.05	8	0.04	104	1	1	0.6	104	6	6	4
2	117	2	40	0.06	8	0.03	107	1	1	0.7	107	7	7	4
3	170	3	25	0.06	8	0.05	150	2	2	1	150	10	10	8
4	223	7	26	0.07	8	0.07	207	4	3	3	207	22	21	19
5	130	3	42	0.08	8	0.05	120	2	1	1	120	9	9	5
6	176	6	43	0.1	8	0.06	160	3	3	2	160	14	13	9
7	245	19	46	0.1	16	0.08	221	7	5	5	221	42	46	36

Table A.39 Results of experiments on the plans for Barman domain(IPC14) found by Probe

Problem	Plan		JUA		IAE		AE		AIAE	UAIAE	GAE		GAIAE	UGAIAE
	Ω	◊	Σ	◊	Σ	◊	Ω	◊	◊	◊	Ω	◊	◊	◊
1	262	21	12	0.1	6	0.1	230	1	1	1	228	14	14	13
2	194	0.5	17	0.1	2	0.1	192	1	1	0.6	192	1	1	1
3	199	2	15	0.1	2	0.1	186	1	1	1	186	4	4	3
4	196	1	14	0.1	2	0.1	194	1	1	1	194	2	2	1
5	180	0.5	15	0.04	2	0.1	178	0.6	0.6	0.6	178	1	1	1
6	199	1	16	0.1	2	0.1	197	1	1	1	197	2	1	1
7	170	0.4	14	0.04	0	0	167	1	0.6	0.5	167	1	1	1
8	161	0.3	15	0.04	0	0	161	1	0.6	0.5	161	1	0.5	0.5
9	229	26	14	0.1	10	0.1	191	1	1	1	191	8	9	8

Table A.40 Results of experiments on the plans for Barman domain(IPC14) found by BFS-F

Problem	Plan		JUA		IAE		AE		AIAE	UAIAE	GAE		GAIAE	UGAIAE
	Ω	◊	Σ	◊	Σ	◊	Ω	◊	◊	◊	Ω	◊	◊	◊
1	199	18	12	0.1	16	0.1	183	1	0.7	0.6	183	6	6	5
2	185	22	13	0.1	16	0.1	168	1	0.7	0.6	168	7	6	6
3	154	0.5	17	0.1	0	0.04	154	0.5	0.5	0.5	154	0.5	0.5	0.5
4	144	1	12	0.4	0	0.04	144	0.4	0.4	0.4	144	0.4	0.4	0.4
5	165	1	15	0.04	2	0.1	162	0.5	0.5	0.5	162	2	1	1
6	153	0.4	16	0.04	0	0.04	153	0.5	0.4	0.4	153	0.5	0.5	0.4
7	173	2	16	0.04	6	0.1	166	0.6	0.6	0.5	166	3	2	2
8	203	9	12	0.1	12	0.1	181	1	1	0.7	179	8	7	6
9	141	1	15	0.03	0	0.04	141	0.3	0.3	0.3	141	0.4	0.3	0.3
10	166	1	13	0.04	6	0.04	160	0.5	0.6	0.5	160	2	2	2
11	178	0.6	16	0.04	16	0.1	162	0.7	0.6	0.6	162	5	5	5
12	168	1	16	0.04	0	0.1	167	0.7	0.7	0.6	167	1	1	1
13	175	3	14	0.04	2	0.1	173	0.7	0.7	0.5	173	1	1	1
14	161	1	14	0.04	2	0.04	159	0.6	0.5	0.4	159	1	1	1
15	159	3	15	0.04	0	0.04	159	0.6	0.5	0.4	159	1	0.5	0.5
16	156	2	12	0.04	0	0.04	156	0.5	0.4	0.3	156	0.5	0.5	0.4
17	142	1	12	0.04	0	0.04	142	0.4	0.4	0.3	142	0.4	0.4	0.3
18	176	2	14	0.04	14	0.1	162	0.7	0.5	0.5	162	5	4	4
19	157	4	14	0.04	0	0.04	157	0.6	0.5	0.5	157	0.6	0.5	0.5
20	160	1	16	0.04	2	0.04	158	0.6	0.5	0.5	158	1	1	1

Table A.41 Results of experiments on the plans for Floortile domain(IPC14) found by Yahsp3

Problem	Plan		JUA		IAE		AE		AIAE	UAIAE	GAE		GAIAE	UGAIAE
	Ω	◊	∑	◊	∑	◊	Ω	◊	◊	◊	Ω	◊	◊	◊
1	52	0.8	14	0.1	8	0	40	0.05	0.05	0.03	40	0.2	0.2	0.1
2	50	0.1	15	0.01	12	0	38	0.04	0.04	0.02	38	0.2	0.2	0.1

Table A.42 Results of experiments on the plans for Floortile domain(IPC14) found by Madagascar

Problem	Plan		JUA		IAE		AE		AIAE	UAIAE	GAE		GAIAE	UGAIAE
	Ω	◊	∑	◊	∑	◊	Ω	◊	◊	◊	Ω	◊	◊	◊
1	46	0	12	0.01	6	0	38	0.1	0.03	0.03	38	0.2	0.2	0.03
2	87	0.02	24	0.03	6	0.02	78	0.3	0.2	0.1	78	2	1	1
3	104	0.1	31	0.03	10	0.03	92	0.3	0.4	0.3	92	3	3	2
4	103	0.04	34	0.04	10	0.02	91	0.5	0.4	0.3	91	3	3	2
5	80	0.02	21	0.04	10	0.01	68	0.2	0.1	0.1	68	1	1	1
6	81	0.02	27	0.02	4	0.01	75	0.2	0.2	0.1	75	1	1	0.5
7	124	0.1	30	0.03	16	0.03	105	0.6	0.5	0.4	105	7	3	4
8	100	0.04	33	0.03	6	0.02	91	0.5	0.4	0.3	91	2	1	1
9	90	0.02	25	0.02	10	0.01	80	0.3	0.2	0.1	80	1	1	1
10	108	0.1	31	0.03	8	0.02	98	0.5	0.4	0.3	98	3	3	2
11	105	0.02	31	0.03	8	0.02	95	0.5	0.4	0.3	95	3	3	2
12	75	0.02	22	0.01	8	0.01	65	0.2	0.1	0.1	65	1	0.5	0.4
13	98	0.04	25	0.02	12	0.01	83	0.4	0.3	0.2	83	3	3	2
14	116	0.04	32	0.03	14	0.02	101	0.6	0.5	0.3	101	4	4	3
15	106	0.04	34	0.03	14	0.02	91	0.5	0.4	0.3	91	4	3	2
16	40	0.1	15	0	2	0	36	0.03	0.03	0.02	36	0.1	0.1	0.1
17	94	0.1	33	0.03	2	0.01	91	0.4	0.3	0.2	91	1	1	0.6
18	122	0.06	30	0.03	2	0.02	101	0.6	0.5	0.4	101	6	5	4

Table A.43 Results of experiments on the plans for Floortile domain(IPC14) found by Probe, BFS-F, and Jasper

Planner	Problem	Plan		JUA		IAE		AE		AIAE	UAIAE	GAE		GAIAE	UGAIAE
		Ω	◊	∑	◊	∑	◊	Ω	◊	◊	◊	Ω	◊	◊	◊
Probe	1	44	81	12	0	4	0	40	0.03	0.04	0.03	40	0.1	0.1	0.1
	2	37	0	15	0	0	0	36	0.03	0.03	0.02	36	0.06	0.06	0.04
BFS-F	1	42	1	12	0.01	2	0.01	38	0.05	0.05	0.4	38	0.1	0.1	0.1
	2	78	98	27	0.03	2	0.02	76	0.3	0.2	0	76	0.5	0.4	0
	3	83	114	24	0.03	4	0.02	76	0.3	0.2	0.1	76	1	1	0.5
	4	38	0.1	14	0.01	0	0	38	0.02	0.2	0.02	38	0.02	0.02	0.02
Jasper	1	44	5	12	0	0	0	44	0.04	0.04	0.04	44	0.06	0.06	0.05
	2	46	2	14	0	6	0	38	0.04	0.03	0.03	38	0.2	0.1	0.1

Table A.44 Results of experiments on the plans for Thoughtful domain(IPC14) found by problem solved by Yahsp3 planner

Problem	Plan		JUA		IAE		AE		AIAE	UAIAE	GAE		GAIAE	UGAIAE
	Ω	◊	Σ	◊	Σ	◊	Ω	◊	◊	◊	Ω	◊	◊	◊
1	46	0.4	14	0.01	0	0	43	0.06	0.07	0.06	43	0.1	0.1	0.1
2	74	0.4	10	0.02	2	0.02	43	0.1	0.1	0.1	43	0.5	0.4	0.4
3	61	0.4	12	0.02	0	0.02	55	0.1	0.1	0.1	55	0.4	0.3	0.3
4	41	0.4	12	0.01	0	0.01	41	0.1	0.05	0.05	41	0.05	0.05	0.05
5	53	0.4	9	0.01	2	0	42	0.1	0.07	0.07	42	0.2	0.2	0.2
6	288	73	7	0.2	0	0	213	2	3	3	238	22	21	21
7	318	0.7	13	0.2	0	0	182	2	3	3	205	35	33	33
8	199	14	12	0.1	0	0	177	2	2	2	177	16	15	14
9	230	14	12	0.1	0	0	171	2	2	2	183	15	15	14
10	246	22	12	0.1	0	0	164	2	2	2	192	18	16	16

Table A.45 Results of experiments on the plans for Thoughtful domain(IPC14) found by Madagascar planner

Problem	Plan		JUA		IAE		AE		AIAE	UAIAE	GAE		GAIAE	UGAIAE
	Ω	◊	Σ	◊	Σ	◊	Ω	◊	◊	◊	Ω	◊	◊	◊
1	34	2	13	0.02	0	0.01	32	0.1	0.1	0.1	32	0.1	0.1	0.1
2	41	3	8	0.02	4	0.01	33	0.1	0.1	0.1	33	0.2	0.2	0.1
3	38	2	15	0.02	0	0.01	34	0.1	0.1	0.1	34	0.1	0.1	0.1
4	48	1	9	0.02	0	0.01	34	0.1	0.1	0.1	42	0.1	0.1	0.1
5	49	12	11	0.03	0	0.02	44	0.1	0.1	0.1	44	0.2	0.2	0.1

Table A.46 Results of experiments on the plans for Thoughtful domain(IPC14) found by Probe planner

Problem	Plan		JUA		IAE		AE		AIAE	UAIAE	GAE		GAIAE	UGAIAE
	Ω	◊	Σ	◊	Σ	◊	Ω	◊	◊	◊	Ω	◊	◊	◊
1	45	0.1	13	0.01	0	0.01	45	0.1	0.1	0.06	45	0.05	0.05	0.05
2	28	0.04	14	0.01	0	0.01	28	0.03	0.03	0.03	28	0.02	0.03	0.02
3	44	0.1	10	0.01	0	0.01	44	0.1	0.1	0.05	44	0.04	0.05	0.04
4	41	0.1	12	0.01	0	0.01	31	0.04	0.04	0.03	31	0.1	0.1	0.1
5	34	0.1	12	0.01	0	0.01	34	0.03	0.03	0.03	34	0.03	0.03	0.03
6	120	4	13	0.1	0	0.1	120	0.5	0.5	0.4	120	0.4	0.4	0.4
7	153	4	14	0.1	0	0.1	146	1	1	1	146	2	2	1
8	157	5	12	0.1	0	0.1	157	1	1	1	157	1	1	1
9	180	11	14	0.1	0	0.1	171	1	1	1	171	4	4	3
10	138	4	13	0.1	0	0.1	138	0.8	0.8	0.7	138	1	1	1
11	190	21	11	0.1	0	0.1	173	2	2	1	173	5	6	5
12	153	14	12	0.1	0	0.1	153	1	1	0.8	153	1	1	1

Table A.47 Results of experiments on the plans for Thoughtful domain(IPC14) found by BFS-F planner

Problem	Plan		JUA		IAE		AE		AIAE	UAIAE	GAE		GAIAE	UGAIAE
	Ω	◊	∑	◊	∑	◊	Ω	◊	◊	◊	Ω	◊	◊	◊
1	29	0	13	0.01	0	0.01	29	0.02	0.03	0.02	29	0.03	0.03	0.02
2	31	0	14	0.01	0	0.01	31	0.02	0.03	0.02	31	0.03	0.03	0.02
3	34	0	14	0.01	0	0.01	34	0.03	0.04	0.03	34	0.04	0.03	0.03
4	32	0	15	0.01	0	0.01	32	0.03	0.03	0.02	32	0.03	0.03	0.02
5	35	0.02	12	0.01	0	0.01	35	0.03	0.03	0.03	35	0.03	0.03	0.03
6	149	4	14	0.1	0	0.1	149	1	1	1	149	1	1	1
7	123	2	13	0.1	0	0.1	121	1	1	1	121	1	1	1
8	143	2	16	0.1	0	0.1	139	1	1	1	139	3	3	2
9	126	4	15	0.1	0	0.1	126	1	1	0.6	126	1	1	0.5
10	173	278	16	0.1	0	0.1	167	1	1	1	167	5	5	5
11	154	5	13	0.1	0	0.1	143	1	1	1	143	4	3	3
12	141	2	16	1	0	1	141	1	1	1	141	1	1	1
13	153	5	14	0.1	0	0.1	151	1	1	1	151	2	2	2
14	146	16	14	0.1	0	0.1	144	1	1	1	144	2	2	2
15	140	11	12	0.1	0	0.1	136	1	1	1	136	3	3	3
16	137	14	13	0.1	0	0.1	137	1	1	1	137	1	1	1

Table A.48 Results of experiments on the plans for Thoughtful domain(IPC14) found by Jasper planner

Problem	Plan		JUA		IAE		AE		AIAE	UAIAE	GAE		GAIAE	UGAIAE
	Ω	◊	∑	◊	∑	◊	Ω	◊	◊	◊	Ω	◊	◊	◊
1	30	0.01	11	0.01	0	0.01	30	0.03	0.04	0.03	30	0.03	0.05	0.05
2	29	0.01	13	0.01	0	0.01	29	0.03	0.03	0.03	29	0.03	0.04	0.04
3	33	0.01	13	0.01	2	0.01	31	0.03	0.04	0.03	31	0.03	0.07	0.06
4	30	0.01	15	0.01	0	0.01	30	0.03	0.04	0.03	30	0.03	0.04	0.04
5	32	0.01	13	0.01	0	0.01	32	0.03	0.04	0.03	32	0.03	0.04	0.03
6	133	1	12	0.1	0	0.1	127	0.6	0.6	0.5	127	2	2	2
7	114	2	11	0.1	0	0.1	111	0.4	0.4	0.4	111	1	1	0.6
8	129	2	11	0.1	0	0.1	114	0.7	0.6	0.5	114	0.6	0.6	0.5
9	129	2	13	0.1	0	0.1	129	0.6	0.7	0.7	129	1	1	0.6
10	197	16	14	0.1	0	0.1	129	0.6	0.6	0.6	180	0.6	0.6	0.6
11	152	1	13	0.1	0	0.1	180	1	1	1	152	8	8	8
12	139	2	11	0.1	0	0.1	152	1	1	0.7	139	1	1	1
13	138	2	16	0.1	0	0.1	138	1	1	0.8	138	1	1	1
14	122	2	12	0.1	0	0.1	122	0.6	0.6	0.8	122	0.6	0.6	0.6
15	130	0.4	13	0.1	0	0.1	130	0.6	0.6	0.6	130	2	0.6	0.6
16	138	2	10	0.1	0	0.1	134	1	1	0.8	134	2	2	2
17	143	109	13	0.1	0	0.1	135	1	1	0.8	135	3	3	3

Table A.49 Results of experiments on the plans for Transport domain(IPC14) found by Probe planner

Problem	Plan		JUA		IAE		AE		AIAE	UAIAE	GAE		GAIAE	UGAIAE
	Ω	◊	∑	◊	∑	◊	Ω	◊	◊	◊	Ω	◊	◊	◊
1	379	50	62	0.2	2	0.1	371	17	16	13	371	66	66	53
2	386	89	53	0.2	8	0.1	331	18	18	15	331	145	144	118

Table A.50 Results of experiments on the plans for Transport domain(IPC14) found by BFS-F planner

Problem	Plan		JUA		IAE		AE		AIAE	UAIAE	GAE		GAIAE	UGAIAE
	Ω	◊	∑	◊	∑	◊	Ω	◊	◊	◊	Ω	◊	◊	◊
1	246	61	56	0.1	14	0.1	230	7	6	5	230	33	39	29
2	294	142	64	0.1	6	0.1	287	10	10	7	287	52	51	38
3	242	82	53	0.1	6	0.1	233	7	6	4	233	32	32	23
4	216	189	58	0.1	2	0.1	211	6	6	4	211	19	19	13
5	315	161	66	0.2	8	0.1	305	13	13	10	305	65	64	49
6	245	41	54	0.2	6	0.1	277	8	7	5	277	39	37	26

Table A.51 Results of experiments on the plans for Transport domain(IPC14) found by Jasper planner

Problem	Plan		JUA		IAE		AE		AIAE	UAIAE	GAE		GAIAE	UGAIAE
	Ω	◊	∑	◊	∑	◊	Ω	◊	◊	◊	Ω	◊	◊	◊
1	195	12	51	0.1	2	0.1	190	4	4	3	190	13	12	9
2	335	67	59	0.2	32	0.1	286	12	11	8	286	160	182	145
3	260	20	52	0.1	22	0.1	234	7	6	5	234	96	89	69
4	282	113	47	0.1	18	0.1	240	9	8	6	240	100	106	84
5	231	38	66	0.2	6	0.1	225	8	7	5	225	30	30	21
6	242	17	52	0.1	16	0.1	222	8	7	5	222	63	62	45

Appendix B

Anytime Planners against Optimization Technique

This appendix presents comparison between Anytime Planners (Lama and Mercury) and pre-optimisation plan technique. It shows different problems of IPC2011 and IPC2014 domains solved by both planners, and first plans optimised by UAIAE. The column 'initial solution' contains plan length Ω and plan generation time $\Diamond$. The column 'Best solution' contains the length of optimised plan Ω and plan optimisation time $\Diamond$, and the column 'UAIAE' contains the length of optimised plan Ω and plan optimisation time $\Diamond$.

Table B.1 Results of experiments on the plans for Barman domain(IPC11) found by Lama

Problem	Initial Solution		Best Solution		JUA		UAIAE	
	Ω	◊	Ω	◊	Ω	◊	Ω	◊
1	157	1	-	-	9	0.04	127	0.3
2	147	0.8	144	1491	10	0.03	124	0.2
3	177	1	167	706	9	0.04	143	0.4
4	154	0.8	-	-	9	0.03	136	0.3
5	162	1	-	-	12	0.04	136	0.3
6	222	4	-	-	10	0.06	156	0.6
6	146	0.4	-	-	11	0.03	132	0.3
7	160	1	-	-	12	0.04	150	0.4
8	188	2	-	-	12	0.05	157	0.5
9	165	1	-	-	13	0.04	157	0.5
10	231	191	-	-	12	0.07	173	0.7
11	208	2	-	-	12	0.06	165	0.7
12	252	9	-	-	13	0.08	197	1
13	231	4	-	-	13	0.07	181	1
14	213	10	-	-	13	0.06	199	0.9
15	225	2	-	-	14	0.07	187	1
16	273	7	-	-	14	0.09	208	1
17	216	1	-	-	13	0.06	190	0.9
18	191	2	-	-	16	0.05	172	0.7

Table B.2 Results of experiments on the plans for Barman domain(IPC11) found by Mercury

Problem	Initial Solution		Best Solution		JUA		UAIAE	
	Ω	◊	Ω	◊	Ω	◊	Ω	◊
1	158	0.4	156	0.8	9	0.03	132	0.3
2	138	0.5	-	-	10	0.03	122	0.2
3	155	0.7	-	-	9	0.03	123	0.3
4	142	0.5	141	1	9	0.03	128	0.2
5	159	0.8	157	1	12	0.04	135	0.3
6	163	1	162	2	11	0.03	145	0.3
7	157	0.8	-	-	11	0.03	137	0.3
8	156	1	154	3	12	0.03	138	0.3
9	202	2	200	5	12	0.05	162	0.5
10	183	1	172	3	13	0.05	161	0.5
11	191	2	189	5	12	0.05	159	0.5
12	196	1	193	3	12	0.05	162	0.5
13	213	1	209	3	13	0.06	181	0.7
14	200	1	198	70	14	0.05	170	0.7
15	204	1	202	3	13	0.05	174	0.7
16	288	5	275	9	12	0.08	215	1
17	232	1	231	3	14	0.06	232	0.9
18	248	2	-	-	14	0.07	198	1
19	222	0.9	-	-	13	0.06	185	0.9
20	165	3	-	-	16	0.05	159	0.5

Table B.3 Results of experiments on the plans for Elevators domain(IPC11) found by Lama

Problem	Initial Solution		JUA		UAIAE	
	Ω	◊	∑	◊	Ω	◊
1	80	0.1	30	0.04	80	0.4
2	143	2	43	0.1	136	2
3	156	2	55	0.1	151	2
4	118	0.6	55	0.08	115	0.7
5	116	0.4	47	0.06	115	0.8
6	184	3	63	0.2	182	4
7	173	2	49	0.1	161	3
8	205	4	66	0.2	202	5
9	198	5	72	0.2	196	4
10	214	3	81	0.3	214	6
11	254	18	81	0.3	241	9
12	265	18	87	0.4	261	8
13	284	64	94	0.4	274	10
14	289	33	104	0.4	281	11
15	288	26	106	0.5	286	11
16	248	20	81	0.7	248	20
17	312	45	91	0.9	300	31
18	313	34	102	1	313	32
19	380	155	107	1	369	45
20	362	57	118	1	358	40

Table B.4 Results of experiments on the plans for Floortile domain(IPC11) found by Lama

Problem	Initial Solution		Best Solution		JUA		UAIAE	
	Ω	◊	Ω	◊	Ω	◊	Ω	◊
1	44	20	35	78	13	0	37	0.02
2	41	19	36	93	12	0	38	0.02
3	57	304	44	881	16	0.01	50	0.05
4	61	229	52	1407	15	0.01	52	0.06
5	51	429	45	720	16	0.01	47	0.05
6	52	424	52	158	17	0.01	48	0.05

Table B.5 Results of experiments on the plans for Floortile domain(IPC11) found by Mercury

Problem	Initial Solution		Best Solution		JUA		UAIAE	
	Ω	◊	Ω	◊	Ω	◊	Ω	◊
1	40	3	35	29	12	0	37	0.03
2	38	3	36	73	12	0	36	0.02
2	52	11	44	140	16	0.01	46	0.04
3	56	444	-	-	16	0.01	52	0.05
4	53	125	45	653	17	0.01	53	0.05
5	56	44	46	467	17	0.01	56	0.06

Table B.6 Results of experiments on the plans for Parking domain(IPC11) found by Lama

Problem	Initial Solution		Best Solution		JUA		UAIAE	
	Ω	◊	Ω	◊	Σ	◊	Ω	◊
1	69	2	37	245	26	0.01	69	0.04
2	46	1	41	3	31	0.01	46	0.02
3	78	6	68	11	28	0.02	78	0.05
4	73	4	-	-	35	0.03	69	0.04
5	64	3	62	4	34	0.01	64	0.03
6	59	5	-	-	34	0.01	59	0.032
7	78	5	42	425	32	0.02	78	0.05
8	70	7	44	1039	37	0.02	70	0.05
9	81	7	58	12	33	0.02	81	0.07
10	90	8	68	13	32	0.02	88	0.06
11	84	9	48	298	43	0.02	84	0.04
12	68	8	-	-	38	0.02	68	0.05
13	77	13	69	34	41	0.02	75	0.05
14	78	18	-	-	36	0.02	78	0.05
15	80	11	74	20	45	0.02	80	0.1
16	87	18	-	-	40	0.02	87	0.1
17	100	15	53	229	39	0.03	100	0.1
18	90	20	51	118	42	0.03	88	0.1
19	92	21	82	45	44	0.03	92	0.1
20	96	23	60	714	48	0.03	94	0.1

Table B.7 Results of experiments on the plans for Parking domain(IPC11) found by Mercury

Problem	Initial Solution		Best Solution		JUA		UAIAE	
	Ω	◊	Ω	◊	Σ	◊	Ω	◊
1	56	2	-	-	26	0.02	56	0.03
2	75	2	42	182	24	0.02	75	0.05
3	65	3	-	-	34	0.02	65	0.03
3	62	3	-	-	36	0.02	62	0.03
4	67	3	40	206	30	0.02	65	0.03
5	65	3	46	1084	31	0.02	65	0.03
6	71	5	-	-	35	0.02	71	0.04
7	82	7	58	9	34	0.02	80	0.05
8	62	5	-	-	38	0.02	62	0.03
9	95	5	46	368	31	0.03	85	0.07
10	71	6	-	-	45	0.02	69	0.03
11	94	10	84	18	37	0.02	92	0.07
12	72	3	58	648	41	0.02	72	0.04
13	83	8	51	1442	40	0.02	81	0.05
13	98	13	58	648	41	0.02	96	0.08
15	76	6	53	987	44	0.02	64	0.04
16	77	12	-	-	46	0.02	77	0.04
17	79	10	67	27	43	0.02	77	0.05
18	85	12	53	630	44	0.02	85	0.06
19	102	27	97	42	39	0.02	102	0.09

Table B.8 Results of experiments on the plans for Scanalyzer domain(IPC11) found by Lama

Problem	Initial Solution		Best Solution		JUA		UAIAE	
	Ω	◊	Ω	◊	∑	◊	Ω	◊
1	14	1.4	10	2	7	0	10	0
2	12	0.2	-	-	12	0	12	0
3	32	0.1	20	313	10	0	32	0
4	40	0.1	26	46	13	0	40	0.01
5	14	0.4	-	-	14	0	14	0.01
6	30	1	14	98	5	0	26	0.01
7	16	1	-	-	16	0.01	16	0.01
8	30	1	18	1556	6	0	26	0.01
9	50	0.2	44	0.7	16	0.01	50	0.04
10	38	0.3	38	0.4	13	0.01	38	0.03
11	60	0.8	42	1639	19	0.01	60	0.01
12	32	0.1	30	1247	11	0	32	0.01
13	26	0.1	24	115	9	0	26	0.01
14	44	0.7	-	-	15	0.01	44	0.04
15	70	1	64	3	22	0.01	70	0.1
16	50	1	-	-	17	0.01	50	0.1
17	80	2	76	5	25	0.02	80	0.1
18	23	36	15	60	12	0	15	0.01
19	79	63	35	254	4	0.01	71	0.01
20	75	47	39	1135	11	0.01	71	0.05

Table B.9 Results of experiments on the plans for Scanalyzer domain(IPC11) found by Mercury

Problem	Initial Solution		Best Solution		JUA		UAIAE	
	Ω	◊	Ω	◊	∑	◊	Ω	◊
1	10	2	10	13	6	0	10	0
2	22	0.2	12	2	22	0.01	20	0.01
3	30	0.02	20	31	9	0	30	0.01
4	36	0.06	26	547	13	0.01	36	0.02
5	26	0.4	14	4	24	0.01	24	0.02
6	30	1	14	94	4	0.01	26	0.01
7	30	0.8	16	9	28	0.01	28	0.02
8	44	1	18	430	6	0.01	40	0.02
9	44	0.1	34	47	16	0.01	44	0.03
10	38	0.2	-	-	13	0.01	38	0.03
11	52	0.4	40	158	19	0.01	52	0.04
12	32	0.1	-	-	11	0.01	32	0.01
13	26	0.06	24	280	9	0	26	0.01
14	44	0.4	-	-	15	0.01	44	0.04
15	60	0.8	52	2	22	0.01	60	0.07
16	50	0.8	48	82	17	0.01	50	0.06
17	68	2	60	5	25	0.02	68	0.1
18	19	36	15	1299	10	0.01	15	0.01
19	59	32	29	335	5	0.01	45	0.04
20	75	26	39	1717	11	0.02	71	0.07

Table B.10 Results of experiments on the plans for Sokoban domain(IPC11) found by Lama

Problem	Initial Solution		Best Solution		JUA		UAIAE	
	Ω	◊	Ω	◊	∑	◊	Ω	◊
1	229	9	157	44	2	0.1	219	2
2	239	2	186	6	0	0.1	239	2
3	129	0.8	115	26	5	0.1	119	0.5
4	309	3	227	21	2	0.1	305	4
5	306	4	200	303	2	0.1	306	3
6	77	0.2	-	-	10	0.1	77	0.1
7	366	21	305	779	3	0.2	342	5
8	433	0.2	429	0.3	2	0.5	433	40
9	279	7	226	43	2	0.1	279	4
10	289	154	187	449	6	0.1	289	4
11	95	4	-	-	4	0.1	95	0.8
12	249	14	175	137	2	0.1	249	3
13	570	62	414	1529	2	0.4	552	12
14	47	0.1	-	-	14	0	47	0.1
15	356	187	-	-	2	0.2	328	5
16	455	561	305	1541	4	0.2	455	9
17	245	523	197	1487	2	0.1	245	2

Table B.11 Results of experiments on the plans for Sokoban domain(IPC11) found by Mercury

Problem	Initial Solution		Best Solution		JUA		UAIAE	
	Ω	◊	Ω	◊	∑	◊	Ω	◊
1	205	4	167	161	2	0.1	195	1
2	189	0.7	198	17	0	0.1	189	2
3	131	0.7	120	5	5	0.05	131	0.5
4	303	6	225	44	0	0.1	299	3
5	423	6	285	47	0	0.2	395	6
6	260	9	196	573	2	0.1	260	2
7	79	0.3	-	-	10	0.03	79	0.1
8	368	29	307	886	3	0.1	348	5
9	433	0.2	429	0.5	2	0.5	433	38
10	291	6	221	502	2	0.1	271	4
11	406	13	-	-	4	0.3	400	12
12	299	472	191	1446	4	0.1	299	4
13	111	303	-	-	4	0.05	109	1
14	243	11	157	102	2	0.1	243	3
15	550	66	348	1090	2	0.3	550	11
16	83	998	75	1458	14	0.06	83	0.4
17	512	163	318	202	2	0.3	492	9
18	737	1461	-	-	4	0.4	617	17
19	277	1465	-	-	2	0.08	277	1

Table B.12 Results of experiments on the plans for Transport domain(IPC11) found by Lama

Problem	Initial Solution		Best Solution		JUA		UAIAE	
	Ω	◊	Ω	◊	∑	◊	Ω	◊
1	191	13	-	-	52	0.1	169	2
2	323	33	-	-	58	0.2	271	9
3	288	11	-	-	50	0.1	204	4
7	336	184	-	-	53	0.5	313	35
5	438	1087	-	-	62	1	402	64
6	336	184	-	-	53	0.5	313	35
7	251	15	-	-	52	0.1	244	6
8	242	28	-	-	62	0.2	218	5
9	212	13	-	-	52	0.1	189	4
10	638	1703	-	-	57	0.5	596	96
11	492	599	482	655	45	0.5	453	53
12	482	377	179		55	0.5	339	38
13	377	567	432	1468	60	1	425	71
14	459	52	348	131	46	0.5	321	36

Table B.13 Results of experiments on the plans for Barman domain(IPC14) found by Lama

Problem	Initial Solution		Best Solution		JUA		UAIAE	
	Ω	◊	Ω	◊	∑	◊	Ω	◊
1	240	2	-	-	14	0.1	189	1
2	221	2	-	-	14	0.1	180	1
3	168	2	-	-	17	0.1	166	0.5
4	258	46	135	1707	12	0.1	184	1
5	204	5	-	-	15	0.1	194	0.5
6	159	2	146	535	16	0.1	157	1
7	218	3	-	-	16	0.1	200	1
8	235	7	-	-	16	0.1	198	1
9	223	7	-	-	12	0.1	191	1
10	262	9	-	-	14	0.1	223	1
11	222	6	-	-	13	0.1	186	1
12	196	7	-	-	15	0.1	179	1
13	310	11	-	-	16	0.1	240	1
14	212	2	-	-	13	0.1	173	1
15	234	1236	-	-	14	0.1	196	1
16	223	6	-	-	14	0.1	181	0.6
17	203	1	-	-	12	0.1	177	1
18	217	2	-	-	14	0.1	185	1
19	238	6	-	-	14	0.1	194	1
20	187	3	-	-	16	0.1	171	0.5

Table B.14 Results of experiments on the plans for Barman domain(IPC14) found by Mercury

Problem	Initial Solution		Best Solution		JUA		UAIAE	
	Ω	◊	Ω	◊	∑	◊	Ω	◊
1	221	1	-	-	14	0.1	185	1
2	233	2	159	1798	13	0.1	183	1
3	168	2	151	1230	17	0.1	162	0.5
4	222	2	136	1230	12	0.1	167	0.6
5	195	5	-	-	15	0.1	177	0.6
6	197	40	-	-	16	0.1	165	0.5
7	196	2	-	-	16	0.1	174	0.6
8	295	4	-	-	16	0.1	220	1
9	252	3	-	-	12	0.1	185	1
10	249	6	-	-	14	0.1	200	1
11	226	2	-	-	13	0.1	188	1
12	200	5	-	-	16	0.1	184	0.6
13	254	34	-	-	16	0.1	200	1
14	228	1	-	-	13	0.1	184	1
15	239	4	-	-	14	0.1	188	1
16	258	48	154	111	14	0.1	212	1
17	201	3	-	-	12	0.1	177	0.6
18	227	2	-	-	14	0.1	187	0.6
19	308	5	-	-	14	0.1	207	1
20	220	2	-	-	16	0.1	184	1

Table B.15 Results of experiments on the plans for Floortile domain(IPC11) found Mercury by Lama and Mercury

Planner	Problem	Initial Solution		Best Solution		JUA		UAIAE	
		Ω	◊	Ω	◊	∑	◊	Ω	◊
Lama	1	39	17	36	78	14	0	38	0.02
	2	39	4	36	5	15	0	36	0.02
Mercury	1	38	0.3	36	28	13	0.01	38	0.02
	2	40	0.2	36	6	15	0.01	38	0.02

Appendix C

Plan Repair via Plan Optimisation

This appendix presents comparison between plan repair and plan optimisation technique. The column 'orginal' contains the length of the original plan generated by agile planner Ω, and the column JAE contains the number of justified unique action in the plan $\sum$. The column 'UAIAE' contains the length of optimised plan Ω and plan optimisation time $\Diamond$, and the column 'LPG Repair' contains the length of repaired plan Ω and plan repair time $\Diamond$.

Table C.1 Results of experiments on the plans for Barman domain(IPC11) found by Probe (invalidated) and repaired by LPG plan repair

Problem	Original	JUA	LPG Repair		UAIAE	
	Ω	∑	Ω	◊	Ω	◊
1	130	9	277	10	128	0.2
2	117	10	200	0.4	117	0.2
3	118	9	188	0.1	118	0.2
4	120	9	156	58	118	0.2
5	144	12	285	60	144	0.3
5	143	11	149	0.1	143	0.3
6	150	11	214	61	150	0.3
7	132	12	189	5	130	0.2
8	159	12	247	22	149	0.4
9	172	13	462	128	166	0.5
10	145	12	308	156	145	0.3
11	155	12	282	88	151	0.4
12	284	12	325	220	224	1
13	187	13	275	97	181	0.7

Table C.2 Results of experiments on the plans for Elevators domain(IPC11) found by Yahsp3 (invalidated) and repaired by LPG plan repair

Problem	Original	JUA	UAIAE		LPG Repair	
	Ω	∑	Ω	◊	Ω	◊
1	262	12	126	3	302	0.6
2	447	29	244	15	630	44
3	661	32	375	38	677	129
4	298	30	21	4	326	20
5	338	22	187	5	346	40
6	776	40	404	45	982	259
7	480	39	267	17	427	429
8	1350	24	562	134	183	1209
9	649	44	402	36	842	748
10	925	43	481	72	194	1211

Table C.3 Results of experiments on the plans for Floortile domain(IPC11) found by Madagascar (invalidated) and repaired by LPG plan repair

Problem	Original	JUA	UAIAE		LPG Repair	
	Ω	∑	Ω	◊	Ω	◊
1	39	15	35	0.02	37	0
2	42	14	36	0.02	86	0.04
3	51	17	44	0.03	51	0
4	54	15	50	0.04	52	0.01
5	50	19	49	0.04	77	0.01
6	52	17	48	0.04	50	0.01
7	75	21	60	0.1	75	0.01
8	64	21	59	0.07	63	0.01
9	91	24	76	0.1	92	0.01
10	74	27	71	0.1	77	0.01
11	85	26	77	0.1	97	0.01
12	86	27	82	0.1	94	0.02
13	118	31	97	0.3	118	0.02
14	120	32	97	0.3	115	0.1
15	132	38	115	0.5	137	0.02
16	131	37	121	0.6	129	0.02
17	178	43	140	1	181	0.1
18	159	43	139	1	157	0.04
19	215	49	169	1	225	0.1
20	182	50	155	1	178	0.1

Table C.4 Results of experiments on the plans for Scanalyzer domain(IPC11) found by probe(invalidated) and repaired by LPG plan repair

Problem	Original	JUA	UAIAE		LPG Repair	
	Ω	∑	Ω	◊	Ω	◊
1	10	8	10	0	10	1200
2	12	12	12	0.01	12	1
3	26	10	26	0.01	36	0
4	32	12	30	0.02	40	1
5	14	14	14	0.01	20	3
6	20	11	20	0.01	24	878
7	16	12	16	0.01	20	9
8	56	2	48	0.03	58	1201
9	40	16	40	0.02	54	0.3
10	38	13	38	0.03	46	1
11	46	19	46	0.03	50	1
12	32	11	32	0.01	34	0.3
13	26	9	26	0.01	32	0.1
14	44	15	44	0.04	54	3
15	54	22	54	0.05	74	2
16	50	17	50	0.06	58	7
17	60	26	60	0.08	70	6

Table C.5 Results of experiments on the plans for Sokoban domain(IPC11) found by Jasper(invalidated) and repaired by LPG plan repair

Problem	Original	JUA	UAIAE		LPG Repair	
	Ω	∑	Ω	◊	Ω	◊
1	177	4	177	0.1	203	0.2
2	201	0	201	2	202	7
3	232	7	218	1	123	19
4	451	0	401	5	453	1
5	473	0	469	7	341	15
6	308	2	294	2	302	0.1
7	90	10	90	0.2	91	3
8	460	3	434	7	365	93
9	445	2	445	37	429	19
10	305	2	267	4	331	0.3
11	612	4	572	21	444	33
13	91	4	91	0.5	101	0.3
14	225	2	221	2	225	0.1
15	586	2	550	10	396	46
16	47	14	47	0.1	49	7

Table C.6 Results of experiments on the plans for Transport domain(IPC11) found by Yahsp3 (invalidated) and repaired by LPG plan repair

Problem	Original	JUA	UAIAE		LPG Repair	
	Ω	∑	Ω	◊	Ω	◊
1	201	27	143	2	269	7
2	230	28	177	3	280	153
3	322	31	214	7	598	94
4	222	21	179	2	249	7
5	207	26	177	2	242	480
6	265	31	211	4	591	9
7	241	36	193	4	395	283
8	435	31	348	19	546	586
9	343	40	269	8	377	866
10	298	37	225	5	454	48