
THE UNIVERSITY
*of* EDINBURGH

# An Investigation into the use of Evolutionary Algorithms for Fully Automated Planning

*Carl Henrik Westerberg*


Doctor of Philosophy

Artificial Intelligence Applications Institute

School of Informatics

University of Edinburgh

2006

# Abstract

This thesis presents a new approach to the Artificial Intelligence (AI) problem of fully automated planning. Planning is the act of deliberation before acting that guides rational behaviour and is a core area of AI. Many practical real-world problems can be classed as planning problems, therefore practical and theoretical developments in AI planning are well motivated. Unfortunately, planning for even "toy" domains is hard, many different search algorithms have been proposed, and new approaches are actively encouraged.

The approach taken in this thesis is to adopt ideas from Evolutionary Algorithms (EAs) and apply the techniques to fully automated plan synthesis. EA methods have enjoyed great success in many problem areas of AI. They are a new kind of search technique that have their foundation in evolution. Previous attempts to apply EAs to plan synthesis have promised encouraging results, but have been ad-hoc and piecemeal.

This thesis thoroughly investigates the approach of applying evolutionary search to the fully automated planning problem. This is achieved by developing and modifying a proof of concept planner called GENPLAN. Before EA-based systems can be used, a thorough examination of various parameter settings must be explored. Once this was completed, the performance of GENPLAN was evaluated using a selection of benchmark domains and other competition style planners. The difficulties raised by the benchmark domains and the extent to which they cause problems for the approach are highlighted along with problems associated with EA search. Modifications are proposed and experimented with in an attempt to alleviate some of the identified problems. EAs offer a flexible framework for fully automated planning, but demonstrate a clear weakness across a range of currently used benchmark domains for plan synthesis.

# Acknowledgements

I would first like to thank my first supervisor, John Levine, for the many insightful and interesting discussions we have had over the years and for all the time and effort he made available for me. I would like to thank my second supervisor Gillian Hayes for her help, especially during the early stages of the PhD.

I would like to thank my examiners Maria Fox and Dave Robertson for their help in improving the thesis.

I am grateful to my funding council, EPSRC, for funding the PhD for the first 3 years. I also thank the School of Informatics, Edinburgh University and my local institute, AIAI, for making funding available to travel to various conferences and workshops. I would like to thank my father for supporting me during the latter stages of the PhD so that I could finish.

I would like to thank the Informatics Computing support staff and the secretarial staff for helping me deal with the practical problems that crop up during a PhD.

Doing a PhD would be a lot less enjoyable and far more time consuming if it was not for my friends and fellow PhD students. Thank you so much!

# Declaration

I declare that this thesis was composed by myself, that the work contained herein is my own except where explicitly stated otherwise in the text, and that this work has not been submitted for any other degree or professional qualification except as specified.

(*Carl Henrik Westerberg*)

For my mother

Dr Valeria Westerberg

November 22nd 1940 - April 20th 2001

# Table of Contents

# Chapter 1

# Introduction

Over the last 10 years domain independent classical planning has experienced very rapid development with new algorithms and new planning domains being investigated [Long et al., 2000b, Bacchus, 2001, Long and Fox, 2003a]. Over the last 20 years EAs have risen to prominence in Artificial Intelligence (AI) and have been recognised as a group of general purpose weak search methods for solving difficult optimisation and machine learning problems, as well as helping to understand the process of evolution [Goldberg, 1989, Mitchell, 1996, Michalewicz, 1999]. The primary research question of this thesis is: "How effective is a basic Evolutionary Algorithm (EA) based planning system at solving domain independent classical planning?" In the field of fully automated planning EA methods have yet to be thoroughly researched when applied to planning, though there have been some promising piecemeal attempts.

This thesis will address the primary question in relation to the problem of plan synthesis. The investigation will be carried out in conjunction with a domain independent EA-based planner called GENPLAN and will cover five main aspects:

1. The behaviour of GENPLAN in relation to how the system is parameterised.

2. An investigation and evaluation of GENPLAN over a broad range of domains and problems once the basic algorithm has been configured.

3. To identify and highlight search issues of the approach that are derived from the classical planning problem and those generated by an EA-based approach.

4. To investigate well motivated improvements to GENPLAN in order to increase the efficiency of planning.

The planning community has always welcomed new ways to search for plans. There are good reasons for why this is the case and this can be most clearly seen with the authors of GraphPlan describing their new planner [Blum and Furst, 1997]:

> "Since searches made by this approach are fundamentally different from the searches of other common planning methods, they provide a new perspective on the planning problem."

I hope that planning by evolutionary search will also open up new perspectives on the problem of planning and introduce a new technique that will prove useful to the planning community.

## 1.1 Planning

Planning is an integral part of a thinking agent that can act rationally. That statement presents planning as two tasks: the first is plan construction and the second is plan execution. A single agent has goals and deliberation or planning is a means of finding a plan to achieve those goals. During plan construction different actions and sequences of actions are explored by knowing an action's expected outcome. Plan execution can occur during or after the planning step and requires the agent to act out the plan. This could be as simple as following a recipe or could require further re-planning if the plan fails in some way. An agent that both builds plans and acts out those plans is deemed to be *situated*.

Planning is a highly desirable facet of human intelligence to replicate in an artificial system. Software systems with greater degrees of autonomy have a chance of being more efficient, safer, and fun. There are environmental situations which are too dangerous for a human to exist but where an artificial agent could solve the problem thus making autonomy desirable. Some examples of practical applications which adopt a top-down approach to planning are: making games more enjoyable and challenging in the card game Bridge [Smith et al., 1998], assisting in search and rescue using I-X[1] [Tate, 2000, Tate et al., 2004], and assisting in the challenges of deep space [Muscettola et al., 1998]. A bottom-up and theoretical approach is also taken for solving planning by the fully automated planning community [Long and Fox, 2003a]. A more basic motivation for studying planning is that rational behaviour is a basic component of intelligence and AI is the study of intelligence.

The ability to think rationally covers a broad spectrum of problems:

[1] http://www.aiai.ed.ac.uk/project/ix/

1. reactive planning: well trained behaviours, for example, following a light
2. simple problem solving: making tea, logistics
3. task ordering: scheduling exams
4. game playing: simple puzzles to video games
5. large scale co-ordinated planning: disaster relief, evacuation, military planning involving many participants and other systems

As has been mentioned, there are two main approaches to planning, termed domain dependent and domain independent planning. A domain dependent implementation that solves the planning problem for a single domain is potentially undesirable. Though domain dependent approaches are the most successful in the field and are used for realistic planning domains, they are costly to implement, only work on one domain (some can operate over more than one domain but there is always a cost of deriving knowledge for a new domain), and obscure any general information about planning that can be derived from them. Usually they are well engineered tools of human intelligence and say little about the more general case, referred to as domain independent planning. Here planning is more abstract and algorithms exist that can solve problems from a variety of domains, though the domains are necessarily simplified models of the world.

In planning terms, the focus of this thesis is on non-situated planning for simple problem solving in a restricted world, but assuming there is a domain independent solution to the planning problem. Domain independent classical planning was chosen as a problem area as it offers the best way in which to begin to assess a new approach to planning. Domain independent planning gives access to a large corpus of domains and problems with which to test a new planner. There also many planners, with a basis in standard AI-planning search, which allows for comparison between new approaches and standard approaches. Classical planning is presented further in Chapter 2

## 1.2 Evolutionary Algorithms

Evolutionary Algorithms are a branch of AI weak search methods that are inspired by nature's evolutionary process. They are weak in sense that they are very general purpose algorithms at a basic level. Examples of the creativity and robustness of evolution are readily available, even to the naked eye. This inspired researchers to "reverse engineer" evolution; to turn it into a computational algorithm. EAs were first used for

optimisation problems and for studying evolution itself, though it is not the case that any synthetic EA claims to be an exact duplicate of natural evolution.

Charles Darwin was the first scientist to recognise the process of evolution and stated four requirements to evolution [Darwin, 1859]:

1. Individuals, within a population, have the ability to reproduce,
2. Differences exist between individuals which affect the likelihood of survival of the individuals,
3. Children inherit properties from their parents,
4. Resources are finite, causing competition.

Those individuals with traits that aid survival in a population will then be more likely to survive. These individuals will have more chance to reproduce and there will be more future individuals with the beneficial traits. This is the main idea behind *natural selection*. EAs simulate the four requirements of evolution. There is a population of individuals, each representing a potential solution to a problem. There is an objective function to discriminate between good and bad individuals. There is a selection function that prefers better individuals. Innovations in new individuals are caused by mutation and recombination which are the evolutionary operators that will also introduce new and useful variations of individuals into the population. This process will then evolve solutions to difficult problems.

The main motivation behind EAs is an attempt to capture some of the creativity and robustness clearly evident in natural systems and to transfer that to a general search algorithm with the same properties. EAs have some desirable properties; due to the population based approach they offer a parallel search and some protection against being trapped in local minima. Also within a population, multiple solutions are simultaneously explored and maintained, this process has been named "implicit parallelism." The main strengths of evolutionary search are that it is easy to implement, generalisable, and can offer innovative solutions. The generalisablity is visible in particular by the multitude of different problems solved by EAs and by the representations used to successfully solve problems. A practical implementation of a basic EA is simple and can be made to solve some quite interesting problems. EAs have already provided some creative and human-beating results[2]. In particular to Genetic Programming, a

[2]http://www.genetic-programming.com/humancompetitive.html

variant of EAs, planning was put forward as good problem for that field, as part of John Koza's motivating argument for the technique [Koza, 1992].

## 1.3 Genetic Planning

The idea of using evolution, fitness evaluation, and a representation based on plans, makes any potential EA-based planning system very flexible. That is, an EA-based planning system can be applied to a number of different tasks within the field of planning. For example a Genetic Planning system can be built for plan synthesis, machine learning tasks, and to optimise plans. Evolutionary techniques have much to offer classical planning.

I focused on the area of plan synthesis as it seemed the most natural area to start from. Unlike standard planning systems, GENPLAN does not reason about its world, but rather simulates various individual plans to discover their fitness and then recombines individuals with high fitness in an attempt to produce an individual of even higher fitness. Because of this, I expect GENPLAN to possibly exhibit different behaviour from standard planners within the planning community.

The most important issues for a genetic planning investigation are:

1. To state the type of planning problem the genetic planner will be applied to.
2. To record how the system is implemented, in particular the plan representation used and the use of any specialised operators.
3. To record the experimental methodology, in particular what statistics will be used and how planning domains are chosen in order to investigate the planner.
4. To present an investigation into the behaviour of the genetic planning system in terms of: parameterisation, performance, search issues, and motivated improvements.

## 1.4 GENPLAN

The primary research question is investigated empirically by experimenting with a testbed genetic planner called GENPLAN. The main design goal behind GENPLAN was to go for simplicity as this makes the best starting point. GENPLAN will be heavily experimented with and it is important that any specialist operators do not obscure

the behaviour of the algorithm. Though GENPLAN is basic it does come with some advantages. Due to the population-based search the algorithm behaves gracefully in terms of memory usage. The planner will not consume all available resources and crash unlike some other fully automated planners. The GENPLAN algorithm is also highly parallelisable and if ever implemented as such would produce massive gains in efficiency.

Having a testbed system has many important uses:

1. GENPLAN will demonstrate what is possible to achieve with a basic EA-based system when applied to the problem of plan synthesis.

2. GENPLAN will be used to identify issues with the approach.

3. GENPLAN will also serve a vehicle with which to investigate improved versions of the algorithm.

## 1.5 Research Questions

The primary research question for this thesis is: "How effective is a basic EA-based planning system at solving domain independent classical planning?" I investigate this question empirically and Cohen gives three basic questions of empirical research [Cohen, 1995]:

1. How will a change in the agent's structure affect its behaviour given a task and an environment?

2. How will a change in agent's task affect its behaviour in a particular environment?

3. How will a change in an agent's environment affect its behaviour on a particular task?

The questions can be rewritten so that they deal with this thesis:

1. How will a change in GENPLAN affect its behaviour given a problem and planning domain?

2. How will a change in the problem affect GENPLAN's behaviour for a particular planning domain?

3. How will a change in the domain affect GENPLAN's behaviour on a particular planning domain?

All the above questions are considered at some point within a specific context. The main research questions this thesis attempts to answer are:

1. "How well have EAs been applied to fully automated planning?"

   Chapter 4 gives a detailed summary of all the available literature where EAs have been applied to various aspects of planning. The chapter pays particular attention to those previous approaches applied to plan synthesis. The results of the survey are that: no single system has had extensive parameterisation experiments performed with it, no single system has been applied to a sufficient range of planing domains and problems, only a limited set of search issues of the approach are presented, and that only a limited set of specialised operators have been used, sometimes with no justification.

2. "How to evaluate a new planning algorithm?"

   Chapters 2, 6, and 8 deal with this issue. There are many sub-issues related to this question.

   The first issue relates to the selection of planning domains. Chapter 2 presents various ways in which domains can be categorised. The most useful of which was to define terms that describe the topology of the search space. This idea had been used previously for describing SAT-instances [Frank et al., 1997]. Defining values for these terms can be used to predict the difficulty of a domain. Over a large set of domains, this analysis can be done to reduce the number of domains necessary to experiment with. This is better than the typical approach in planning which is to evaluate a planner against a set of domains without considering how the domains may affect the behaviour of the search algorithm. Chapter 6 gives a first attempt at analysing domains in terms of the heuristic used by GENPLAN.

   The second issue relates to the choice of problems for the domains. It is very important to use a good random set of well defined problem instances. An evaluation can be improved by also incorporating known problem instances but they too should be of good quality.

   The third issue of improving an evaluation of a planner is to incorporate results from a set of planners built around standard planning technology. The results of the various planners are used to assess the scaling behaviour of the new planner.

Chapter 6 deals with pragmatic concerns of running the actual experiments and deciding on what results to present. Metrics should be chosen that can assess both the efficiency of a planner and the quality of plans produced. The experimental methodology should include all decisions relating to how the planner was run and the environment it was run in.

3. "How to implement an EA-based approach to fully automated planning?"

   Chapters 3, 4, and 5 take this question into consideration. Chapter 3 deals generally with the problem of describing the basic components of an EA. Chapter 4 deals with the more interesting case of summarising different approaches made with previous implementations of EA-based planning systems. The most important issues are: the representation of plans in the implementation, the use of any domain dependent knowledge to aid search, and a description of any non-standard components used. Chapter 5 gives exact details of the implementation of GENPLAN.

4. "How best to parameterise an EA to solve planning?"

   Chapter 7 describes the behaviour of GENPLAN using a variety of parameterisation experiments. The results are then used to set the values of the various parameters. The general conclusion is that different parameterisations can have an effect on the performance of GENPLAN but not to the extent that great performance gains can be made through parameterisation alone.

5. "How well does GENPLAN perform?"

   Chapter 8 presents results of the investigation and evaluation of GENPLAN over a carefully selected range of domains and problems. The evaluation is completed by running a set of planners on the same problems, and incorporating an examination of scaling behaviour.

6. "What properties of search make planning problems difficult for an EA based approach?"

   Chapter 9 relates the search issues raised in Chapters 2, and 6 to the efficiency of GENPLAN using pathological examples. The behaviour of GENPLAN in the face of such issues needs to be explored as it will go some way to describing the behaviour of GENPLAN.

7. "What problems are faced with taking an EA-based approach to planning?"

   Chapter 9 also deals specifically with problems caused by an EA-based approach. The issues briefly examined are: diversity, bloat, and crossover. Diversity relates to how different individuals present in a population are from one another. Bloat is the issue of growth of individuals during a run of an EA system. Finally crossover is a controversial operator and the behaviour of one type of crossover is examined.

8. "How can the problems of an EA approach be alleviated?"

   Chapter 10 presents a series of promising improvements to GENPLAN. These improvements are geared towards dealing with some of the search issues raised in Chapter 9.

## 1.6 Organisation

The thesis should be read linearly with the organisation of the thesis given as follows:

- Chapter 2: Presents the classical planning problem in conjunction with the representation language used, characteristics of planning domains are summarised, and standard approaches for tackling planning are presented.

- Chapter 3: Provides an introduction to the field of Evolutionary Algorithms, paying particular attention to Genetic Algorithms and Genetic Programming. The chapter presents the basics to theoretical considerations of the techniques.

- Chapter 4: Gives a detailed summary of all the available literature where evolutionary techniques have been applied to various aspects of planning. The chapter details all EA-based approaches to the plan synthesis problem.

- Chapter 5: Gives a detailed presentation of GENPLAN describing how each functional component has been implemented. The chapter summarises how GENPLAN differs from normal GAs and GP and from the previous approaches to EA-based plan synthesis.

- Chapter 6: The chapter is divided into two parts. Firstly a summary of statistics and hypothesis testing is given. Secondly the chapter presents the experimental methodology adopted by the thesis and characterises the planning domains used for investigating and evaluating GENPLAN.

- Chapter 7: Summarises results from many different parameterisations of GENPLAN over a limited set of domains and problems in order to find an acceptable parameterisation of GENPLAN. The parameterisation experiments also explore the behaviour of GENPLAN.
- Chapter 8: Investigates and evaluates GENPLAN on a variety of classical planning domains. The evaluation is performed in conjunction with a set of test planners.
- Chapter 9: Highlights the issues with using an EA-based approach to performing classical planning that can be derived from both the planning problem and issues resulting from an EA-based approach.
- Chapter 10: Presents experiments highlighting improvements I have implemented in an attempt to address the issues identified in Chapter 9 and thereby improve the efficiency of GENPLAN.
- Chapter 11: Finally, the thesis is finished with a reconsideration of the research questions.

# Chapter 2

# Fully Automated Planning: Representation, Domains, and Search

Automated AI planning is the task of transferring the process of planning to the computer. The first step is to represent the world in a means that is understandable to both the computer and practitioner. The representation also describes the planning problem, in this case classical planning. Planning domains are created using the representation that represent interesting problems. The second step is to then implement an algorithm that can accept this representation, understand it, and then devise plans based on the available actions that achieve the goals of the problem. With an internal representation of the world a search algorithm can deliberate over the effects of various actions. Standard approaches to planning will be presented in this chapter. The third step is to then explain the differing computational effort required by various planning algorithms over a range of different domains. This final step is necessary as there is a need in the field to mature and to develop a methodology for the design of planning algorithms [Hoffman, 2003]. The behaviour of different planning algorithms can potentially be explained by analysing the characteristics of the planning domains and relating back the characteristics to the search algorithm employed.

## 2.1 The Classical Planning Problem

Fully automated planning systems make several simplifying assumptions about the world in order to ease both representation and computation requirements. The world is discretized so that everything about a world can be described by its state. The state of the world is everything that is known by the planner.

The key assumptions are [Ghallab et al., 2004]:

- Finite set of states: Given a planning problem it is possible to enumerate all possible states. This means that objects cannot be created or destroyed.
- Fully observable: A planner has full knowledge of the state, no information is hidden from the planner.
- Deterministic: An action will always succeed or fail when it is meant to. The effect of an action is always the same.
- Static: Only the planner can affect changes to the state via actions. No external forces can affect change on the state.
- Restricted goals: A more minor assumption in that only a goal state or set of goals states are specified. No additional constraints can be added to a problem, for example states to be avoided.
- Sequential plans: Plans are made up of an ordered list of actions. An extension would be to allow actions to happen in parallel.
- Implicit time: All state transitions happen instantly and there is no concept of actions taking a certain amount of time.
- Offline planning: A planner finds a plan unconcerned about what is occurring in the world while it is planning. It is a assumed the plan will be executed at a later time and still be valid.
- Single agent: Only one agent exists in the world and only this agent can alter the state.

This is a highly restricted view of the world, but forms the basis of most modelling languages. These assumptions lend themselves to propositional and first order logic-based representations of the world. Even with this restricted representation of the world, reasoning can still be very difficult, see Section 2.2.1 for more details. Much of the work in the field of fully automated planning goes into modifying or devising new algorithms that can handle domain independent planning with these assumptions and relaxations to the assumptions.

### 2.1.1 STRIPS

The STRIPS representation language [Fikes and Nilsson, 1971] formalises the representation of planning domains and adopts many of the assumptions stated in Section 2.1. STRIPS will be used to introduce the classical planning problem as STRIPS-like planning forms the basis of most planning problems and those used within this thesis.

#### 2.1.1.1 A Planning Problem

The planning problem will now be described more formally in terms of STRIPS. A planning problem, $P$, is represented with the tuple: $P = (A, O, I, G)$, where

- $A$ is the set of atoms
- $O$ is the set of operators
- $I$ is the initial set of atoms, $I \subseteq A$
- $G$ is the goal set of atoms, $G \subseteq A$

A STRIPS problem is further defined as:

- Having a state $s$, where $s \in S$, $S$ is the set of all states and $s$ is a collection of atoms from $A$.
- $s_0$ is the initial state $I$ and $s_g$ is the goal state $G$, with $s_0, s_g \in S$.
- An action $a$ is an instantiation of an operator $o$, with $o \in O$. An action is made up of three parts: the preconditions, $Pre(a)$, the add list, $Add(a)$, and the delete list, $Del(a)$. Each part is made up of a set of atoms.
- A transition function, $f(s,a)$, describes how states are updated by actions. If $s_c$ is the current state, $a$ is the action being examined, and if $Pre(a) \subseteq s_c$, then the transition function is: $s' = f(s,a)$ such that $s' = s - Del(a) + Add(a)$.

#### 2.1.1.2 Atoms

Atoms are made up of predicates and objects. A predicate defines a relation about the world. For instance, `on(?x ?y)` can be used to describe that an object is on top of another object. A predicate, plus objects, make up an instantiated atom. Atoms are also referred to as propositions.

#### 2.1.1.3 Actions

An action is an instantiation of an operator. Each operator has a unique name and a list of parameters. The parameters are used to instantiate the predicates in the three parts of the operator: the precondition list, the add list, and the delete list. The `move-briefcase` operator from the `briefcase` domain takes three parameters: `?brief` defining which briefcase to move, `?location1` where the briefcase currently is, and `?location2` the location to move the briefcase to. For instance, `move(briefcase1, london, paris)` is an example of an action. It has the following lists:

- $Pre(move(briefcase1, london, paris))$ = `[at(briefcase1,london)]`
- $Add(move(briefcase1, london, paris))$ = `[at(briefcase1,paris)]`
- $Del(move(briefcase1, london, paris))$ = `[at(briefcase1,london)]`

In order for an action $a$ to update a state $s$, all the atoms in the precondition list must be present in $s$, $Pre(a) \subseteq s$. If this is the case a new state $s'$ is constructed by adding all atoms in $Add(a)$ to $s$ and deleting all the atoms in $Del(a)$ from $s$.

Given problem with a set of objects, it is possible enumerate all possible actions for a particular domain. Let $A_{all}$ be the "action space" which is the set of all actions formed by permuting all possible combinations of parameters to every operator in the domain. Let $A_s$ be the set of all actions that apply in state $s$: $a \in A_s$ if $Pre(a) \subseteq s$.

#### 2.1.1.4 States

States are made up of atoms. Only atoms present in the state are known to be true and all other atoms are assumed to be false, which follows from the closed world assumption. The two most important states are $s_0$ and $s_g$: $s_0$ being the initial state, and $s_g$ being the goal state. The current state is denoted by $s_c$. Continuing with the `briefcase` domain an example state is:

```
[at(obj1, loc1),   at(briefcase1, loc1),
 at(obj2, loc2),   at(briefcase2, loc2),
 at(obj3, loc3),   at(briefcase3, loc3)]
```

#### 2.1.1.5 Plans

Given $P = (A, O, I, G)$, a plan $p$ is a list of actions $[a_1, a_2, ...a_n]$ that when executed transforms $s_0$ to $s_g$, or a state $s$ that has $s_g$ contained within it. A sequential plan

defines a linear series of state transitions where each action is executed in turn one after another, or in a "total-order." For a given domain and problem it is expected that several plans exist that can solve the problem. In addition, there is a notion of optimality for plans. For STRIPS planning with sequential plans the optimal plan for a problem is the one that achieves the goal state using the fewest number of actions. Sub-optimal plans achieve the goal state by using more actions than are strictly necessary.

#### 2.1.1.6 An Example from `maxones`

In order to make the above statements more tangible an example problem is given from the `maxones` domain. This domain is simple and it is possible to list all the features of STRIPS planning very easily.

In this domain there are a set of $N$ switches which are initially off and must be switched on. There is one operator, $switch(?x)$ which turns a switch from off to on. The goal is to find a plan that switches on all the switches. Using the definitions:

- A = off(switch1), off(switch2), on(switch1), on(switch2)
- O = switch(?x): $Pre$[`off(?x)`] $Add$[`on(?x)`] $Del$[`off(?x)`]
- $s_0$ = [`off(switch1)`, `off(switch1)`]
- $s_g$ = [`on(switch1)`, `on(switch2)`]

A plan that solves this problem is:

1. switch(switch1)
2. switch(switch2)

### 2.1.2 PDDL

PDDL stands for Planning Domain Definition Language, the first version of which was created for the first International Planning Competition (IPC), held during the fourth International Artificial Intelligence Planning and Scheduling conference (AIPS-98) [Long et al., 2000b]. The motivations behind PDDL are to create a planning language that is easily extendable, that would facilitate the development of a standard suite of benchmarks, and to allow more empirical comparison between different planners. Before the development of PDDL researchers relied on their own specialised languages and domains making any kind of comparison of planning technology difficult.

PDDL is important as it implements a flexible framework allowing for ever increasing complexity of planning domain. Any researcher that develops a planner supporting PDDL gives the planner access to many standard domains and problems. In addition, this allows for effective comparison of systems in terms of capability, efficiency, and plan quality. Together, PDDL and the biennial planning competitions have helped draw interest and encouraged progress in the field of planning.

#### 2.1.2.1 PDDL 1.2

The first implementation of PDDL included many features, some of which were never used for the competitions. PDDL 1.2 [McDermott et al., 1998] includes the following features:

- Basic STRIPS-style actions
- Conditional effects
- Universal quantification over dynamic universes (supporting object creation and destruction)
- Domain axioms
- Safety constraints
- Hierarchical actions
- Mathematical expressions and fluents
- Management of multiple problems in multiple domains using differing subsets of language features

The scope of the language was extremely ambitious, and for the first competition not all the elements of language were required. The first set of domains used for IPC 1 were written in a language equivalent to STRIPS though some included typing information.

#### 2.1.2.2 PDDL 1.3

For IPC 2 [Bacchus, 2001], it was decided to use a subset of the PDDL language so that the competition could be as inclusive as possible. The features that survived for this competition are [Bacchus, 2000]:

- STRIPS-style actions
- Typed objects
- ADL (subsumes STRIPS and allows the following constructs: disjunctive preconditions, equality, existential and universal preconditions, and conditional effects)

#### 2.1.2.3 PDDL 2.1

In PDDL 2.1 the language has been updated to allow for the definition of new and more complex planning domains, specifically to better support numeric values, and to begin supporting actions with durations and continuous effects [Fox and Long, 2001a]. The motivation for this is to apply existing planning technology to more realistic domains. This extension to the language represents a major change in fully automated planning. Some of the planners entered into the third competition coped admirably with the new complexities [Long and Fox, 2003a]. The language extensions are divided into five levels:

- Level 1: STRIPS
- Level 2: Numeric Variables which allow testing and updating of values
- Level 3: Durative actions with no continuous effects
- Level 4: Durative actions with continuous effects
- Level 5: A way forward for the design of an expressive planning domain description language capable of representing arbitrary real-time mixed discrete and continuous domains

#### 2.1.2.4 PDDL 2.2

The latest version of PDDL makes a modest contribution to PDDL. The previous extensions formed a large change to the language and issues of resources and time need to be further explored by the community. The results from the latest competition have not yet been formally published. The two minor extensions to the language are [Edelkamp and Hoffman, 2003]:

- Derived predicates: Rules which allow updates to state without using actions, for example modelling flow. This extension formalises the axiom rules of PDDL 1.2.

- Timed initial literals: Allows facts in a state to become true or false at specific times.

## 2.2 Planning Domains

An open topic of research is to be able to classify the various planning domains into categories based on attributes of interest. This is important in the interests of benchmarking so that planning authors can test for known categories and search issues. It is important to know how domains function in relation to search so that results are not presented from a series domains that are in fact very similar or which cater to an approach. Domain theory is also important for constructing a theory of planning in order to predict when a particular algorithm will perform well on a certain class of domains.

Currently, there are a number of ways to categorise domains:

1. Categorising domains based on various known complexity results [Bylander, 1994, Helmert, 2003].

2. Categorising domains against the local search space formed when using the relaxed planning heuristic $h^+$, as defined by [Hoffman, 2003].

3. The third measure is to identify generic types within domains [Long and Fox, 2000, Long and Fox, 2001, Murray, 2003].

4. A fourth and practical consideration is to include which subset of PDDL is required to run the domain.

5. A fifth category is to chart their appearance in the planning competition.

### 2.2.1 Complexity of STRIPS-Planning

Bylander contributed the first major and contiguous analysis of the complexity of planning within propositional STRIPS [Bylander, 1994]. Bylander examined the worst case complexity of STRIPS-based planning by altering the syntax of various domains. In the worst case this type of planning is *PSPACE*-complete. The syntactic variations

| Domain | Plan Existence | Bounded Plan |
|---|---|---|
| Logistics | *polynomial* | **NP**-*complete* |
| Gripper | *polynomial* | *polynomial* |
| Blocksworld-noarm | *polynomial* | **NP**-*complete* |
| Miconic-STRIPS | *polynomial* | **NP**-*complete* |
| Movie | *polynomial* | *polynomial* |
| Grid | *polynomial* | **NP**-*complete* |
| Schedule | *polynomial* | *polynomial* |
| Mystery | **NP**-*complete* | **NP**-*complete* |
| Mprime | **NP**-*complete* | **NP**-*complete* |
| Miconic-ADL | **NP**-*complete* | **NP**-*complete* |
| Freecell | **NP**-*complete* | **NP**-*complete* |

Table 2.1: Complexity Results for various benchmark domains

are of the type: number of preconditions, number of post conditions, (allowing negations in pre or post conditions), and in one specific case the number of goals. One of the most notable results to come out of this work is the recognition that bounded (optimal) planning for domains that contain operators with any number of preconditions and any sized add lists is still **NP**-*complete*. This has important implications for the calculation of the relaxed planning heuristic $h^+$.

The results in the previous paragraph are general and the next step is to examine domains individually [Helmert, 2003]. As Bylander's results only apply to certain syntactic situations it maybe that an actual domain presents different behaviour because of structure in the problem. The main results of Helmert's paper are summarised in Table 2.1. Plan existence is the task of finding any plan to solve a problem. Bounded planing is the problem of attempting to find a plan within a specific length and corresponds to optimal planning. It is important to note that for some domains polynomial time algorithms exist for finding plans, but some planners still demonstrate exponential behaviour on those domains. As will be shown in the next subsection, complexity alone is not enough to differentiate between domains, or even problems from a domain.

### 2.2.2 The $h^+$ Categories

The $h^+$ heuristic is a commonly used heuristic estimate for rating planning states during search. The heuristic returns an estimate of the number of steps required to transform $s_c$ to $s_g$. The heuristic estimate is derived by solving a simplified planning problem, one in which delete lists of actions are ignored. Even solving this relaxed problem optimally is **NP**-*complete* [Bylander, 1994], so any heuristic value solving the simplified problem is an estimate. Hoffman has used this heuristic very capably with the FF planning system [Hoffmann and Nebel, 2001] which has subsequently lead the author to investigate the search structure of planning problems in relation to this heuristic. The $h^+$ heuristic is described in more detail in Section 2.3.5.1 as this section concentrates on using the heuristic to characterise planning domains. The categories are also used to predict and characterise the performance of GENPLAN.

Hoffman defines two axes for describing the search topology of domains in terms of the $h^+$ heuristic [Hoffman, 2003]. The first axis relates to *dead-ends*. A dead-end occurs when at $s_c$, $s_g$ is no longer reachable. A good heuristic should return an estimate of infinity for that state, $h(s_c) = \infty$. The second axis relates to the local search topology around $s_c$. The author defines a number of different shapes the topology can take around a point $s_c$ and several categories of dead-ends. This work adopts the assumption that lower heuristic values are better than higher heuristic values. Local search topology has been looked at in relation to solving SAT problems [Frank et al., 1997] and in general search topologies are well studied problem in other areas of AI [Langdon and Poli, 2002].

- Dead-end categories: listed from easiest to hardest
    - *undirected:* This is the simplest case and occurs when the planning domain is fully reversible which means that any state is reachable from any other state.
    - *harmless:* This is the next most difficult, actions are no longer fully reversible but no irreversible action makes $s_g$ unreachable.
    - *recognised:* The third most difficult case is when there exists states such that $s_g$ is no longer reachable from $s_c$ but the heuristic correctly recognises all such states.
    - *unrecognised:* The final category is an extension to the previous category in that the heuristic now fails to recognise some states as dead-ends.

- Local search terminology
  - *level:* This is the heuristic value for the current area.
  - *plateau:* This is an area in the search space where the heuristic returns the same value for all states in the plateau.
  - *exit:* A state transition to a new state, where the heuristic value of the new state is different to the previous state.
- Local search surface topology
  - *recognised dead end:* This is a plateau of level $\infty$.
  - *global minimum:* A plateau of level 0.
  - *local minimum:* There are no exits where $h(s_{exit}) < currentlevel$.
  - *bench:* There is at least one exit from the plateau which allows search to reach a lower value than the current level.
  - *contour:* This is a plateau that consists entirely of exits.

The biggest difficulties for a search algorithm are local minima and benches. In the case of local minima it is important to know the number of steps that have to be made in order to reach an exit that could lead to a global optimum (backtracking). In the case of benches it is important to know the size of the plateau, or the expected length of the flat path in order to exit the plateau. Hoffman defines two abbreviations for the two cases:

- *mlmed* (*maximal local minimum exit distance*): the largest exit distance from all states on the local minima.
- *mbed* (*maximal bench exit distance*): the largest exit distance from all states on the plateau.

Table 2.2 categorises a number of benchmark domains against the two metrics. The letter $c$ is a constant and can be calculated exactly for some domains. For some domains and heuristics the exact value can vary depending on how problems are defined. The most important part of this research is that these metrics are general and can be reformulated for other heuristics. The metrics can be used to predict that the easy domains are near the top of the table and the harder domains are near the bottom end of the table.

| Domain | Dead-end Cat ($h^+$) | Topology ($h^+$) |
|---|---|---|
| Logistics | undirected | mbed $<= c$ |
| Ferry | undirected | mbed $<= c$ |
| Gripper | undirected | mbed $<= c$ |
| Hanoi | undirected | mlmed $<= c$ |
| Blocksworld-noarm | undirected | mlmed $<= c$ |
| Fridge | undirected | mlmed $<= c$ |
| Briefcaseworld | undirected | mlmed $<= c$ |
| Blocksworld-arm | undirected | mlmed unrestricted |
| Depots | undirected | mlmed unrestricted |
| Driverlog | undirected | mlmed unrestricted |
| Tireworld | harmless | mbed $<= c$ |
| Satellite | harmless | mbed $<= c$ |
| ZenoTravel | harmless | mbed $<= c$ |
| Miconic-SIMPLE | harmless | mbed $<= c$ |
| Miconic-STRIPS | harmless | mbed $<= c$ |
| Movie | harmless | mbed $<= c$ |
| Simple-TSP | harmless | mbed $<= c$ |
| Grid | harmless | mlmed $<= c$ |
| Schedule | recognised | mbed $<= c$ |
| Rovers | recognised | mlmed unrestricted |
| Mystery | unrecognised | mlmed unrestricted |
| Mprime | unrecognised | mlmed unrestricted |
| Miconic-ADL | unrecognised | mlmed unrestricted |
| Freecell | unrecognised | mlmed unrestricted |
| Assembly | unrecognised | mlmed unrestricted |

Table 2.2: Categorisation of domains using $h^+$

### 2.2.3 Generic Types

Once domains are examined more closely a planning practitioner starts to realise that planning domains fall into definite classes. Domains from a particular class appear almost equivalent but can differ in some way, for example the `gripper` domain uses a restricted resource to carry objects around, whereas the briefcase in the `briefcase` domain has no limit to the number of objects it can carry. There are a number of generic classes of domain that have been identified: transportation domains [Long and Fox, 2000], constructions domains [Clark, 2001], and resource allocation domains [Long and Fox, 2001]. The original purpose behind identifying ways in which to classify domains was to augment a planner called STAN with specialised sub-solvers [Fox and Long, 2001b].

The final point to make about generic types is that they are general and can be used to create control information to assist in search [Murray, 2003]. So that it may be possible for a generalised planner to exist that identifies key parts of the problem which customises or augments itself for a particular application.

### 2.2.4 Level of PDDL

Each level of PDDL builds on the preceding level and designers have been careful to ensure that new levels are backwardly compatible with preceding levels. The most frequently used levels of PDDL, given by their PDDL requirement flag, are:

- `:strips`, specifies STRIPS-style actions as explained in this chapter.
- `:typing`, this allows objects in a domain to have types. It serves to make domains more readable and is an aid to search.
- `:adl`, this encompasses a number of language extensions: negative preconditions, disjunctive preconditions, equality, quantified-preconditions, and conditional effects.
- `:fluents`, this declaration is in reference to Level 2 of PDDL 2.1 (see section 2.1.2.3)
- `:durative-actions`, allows actions to have a duration.
- `:continuous-effects`, allows actions that take a duration to have an effect on a fluent continuously over a period of time.

| IPC 1 (1998) | IPC 2 (2000) | IPC 3 (2002) | IPC 4 (2004) |
|---|---|---|---|
| Gripper | Blocksworld-4ops | Depots | Airport |
| Movie | Logistics | DriverLog | Pipesworld |
| Logistics | Schedule | Zenotravel | Promela |
| Mystery | Freecell | Satellite | PSR |
| Mystery′ (Mprime) | Miconic | Rovers | Satellite |
| Grid | | Freecell | Settlers |
| | | Settlers | UMTS |

Table 2.3: Domains categorised by their appearance in the competitions

- `:derived-predicates`, allows external establishment of propositions without using an action.
- `:timed-initial-literals`, allows propositions to become true at specific times.

Domains can be classified along these levels. This serves a useful practical purpose to identify domains that are compatible with certain planners as not all planners support all extensions.

### 2.2.5 Competition Domains

Table 2.3 presents the domains used to test fully automated planners as they have appeared for each competition. The domains are of ever increasing complexity, with those used in the last competition being the most difficult and are the best at simulating real-world problems.

## 2.3 The Main Search Techniques

This section summarises different search-based approaches to solving fully automated planning. Planning algorithms can be defined across a number of attributes:

1. A definition of the search space being searched.
2. The way in which the plan is refined from the search space.
3. The type of plan produced, in terms of ordering constraints.

4. Whether the plan produced is optimal or sub-optimal.

### 2.3.1 State-Based Search

State-based search is the simplest and most immediate technique for solving planning problems. From a particular state $s$ it is possible to find all possible actions that apply in $s$, $A_s$, $a \in A_s$ *if* $Pre(a) \subseteq (s)$. State-based search starts at $s_0$ and possible future states are found, by finding $A_s$ and applying each action in that set. The resulting states are recorded and searched over. There are a number of different orders in which to search the newly generated states. In terms of graphs, states represent nodes, and actions represent arcs between nodes. In this way, state-space search techniques partially map a state space looking for a solution. In planning, entire search spaces cannot be explored as they tend to grow exponentially.

The simplest kind of search is a random walker search. An algorithm in pseudo code is presented in Algorithm 1. The algorithm is sound, in that all plans the algorithm returns are valid. The algorithms for state space search return totally-ordered sequential plans. The order the state space is searched determines whether a plan returned by the algorithm is optimal or not, for example the difference between Breadth First Search and Depth First Search.

There are a couple of benefits of applying state-based search to planning. Firstly it helps teach new students to the planning field the basics of planning. Secondly algorithms based on state-spaced search are quite easy to implement. However due to the way state spaces explode during search they are impractical for larger problems. The inherent difficulties of state-based search have lead researchers to develop sophisticated pruning techniques, sophisticated heuristics, and to create different search spaces in which to find plans. Heuristic search has become one of the main techniques for tackling the planning problem. Currently, there are no modern planners that use pure state-based search.

### 2.3.2 Partial Order Planning

Partial Order Planning (POP) turns the search space for planning on its head when compared with state-based search. Rather than deriving a plan from the state space, search is performed in the plan space using partial plans. An excellent tutorial to POP and to planning as a whole can be found in [Weld, 1994]. A total-order planner based on state search has to make explicit decisions at what point actions occur and that

**Algorithm 1** An algorithm for finding plans using simple search

1: $s = s_0$
2: $p = []$
3: **while** $s \subseteq s_g$ **do**
4: Find $A_s$ for $s$
5: Select $a$ randomly from $A_s$
6: $s = f(s,a)$
7: Add $a$ to $p$
8: **end while**
9: Return $p$

the actions cannot act in parallel. Whereas in POP, actions have ordering relations specified between them. This means that a partial plan represents a set of possible plans. POP is also based strongly on the idea of *least commitment*. This rationale is to leave restrictive decisions right up until the last moment, as it could be beneficial to delay those decisions. As an analogy to state-space search the plans form nodes in the space and operators that refine a plan form arcs to other plans.

More formally, POP is made of up four components: $A, O, L$ and threats:

- $A$ is the current set of actions in the plan. The set is initialised with two special actions at the beginning of search: $A_0$ and $A_\infty$. $A_0$ has no preconditions and its effects are simply the initial state of the problem, likewise $A_\infty$ has no effects and its preconditions are the goal state .

- $O$ is the current partial set of ordering constraints on the actions. An ordering constraint specifies which actions must occur before others. $O$ is initially assigned $A_0 < A_\infty$. The constraints in $O$ must be maintained so that there is at least one valid total-order plan that satisfies all the constraints.

- $L$ is the set of causal links. A causal link can be defined informally as a means of recording decisions made during the planning process. That is, which actions are used to support the propositions in the problem. An abstract example of a causal link is: $A_p \xrightarrow{Q} A_c$. $Q$ is a proposition that is an add effect of $A_p$ and a precondition of $A_c$.

- A *threat* occurs when an action deletes the precondition of another action. This can be checked by looking at the causal links and the current ordering constraints

on the actions. It is similar to the concept of *mutexes* that will be introduced in the graphplan section. There are two ways of resolving threats:

1. *promotion:* Constrain the action $A_{threat}$ to come after $A_c$, $A_c < A_t$.
2. *demotion:* Constrain the action $A_{threat}$ to come before $A_b$, $A_t < A_p$.

From the above it is possible to build up the basic POP algorithm. The final element to add is the agenda. The agenda contains the propositions that have not yet been supported by an action. As the basic POP algorithm is regressive the preconditions of $A_\infty$ are added first. The ordering constraints, actions, and links are initialised as above. A proposition Q from the agenda can be supported either by a new action $A$ or an action that has already been chosen. If an action is reused then a consistent ordering must exist. If the action is a newly instantiated action then its preconditions are added to the agenda. The causal links are then checked and if a threat is detected then either promotion or demotion is applied. The algorithm continues until the agenda is empty and there are no threats.

#### 2.3.2.1 UCPOP

UCPOP [Penberthy and Weld, 1992] was the foremost planner to implement the POP algorithm. UCPOP makes a number of extensions to the basic POP algorithm, mainly concerned with handling more complicated language specifications, like ADL. The original UCPOP paper then goes on to show UCPOP being able to solve simple problems from the `briefcase` and `blocksworld` domains.

One of the major benefits of POP is that the returned plan is partially ordered. This gives the plan a high degree of flexibility which is useful for when plans are executed. But partial order planning is no longer as actively researched as it once was due to the success of new search techniques. One of its drawbacks is that it ignores explicit information available in the state, in which new heuristic planners can take advantage of.

After UCPOP was released the planning community invested many years in trying to improve the POP algorithm by giving it better flaw selection strategies (the order in which to deal with threats and unsupported preconditions as examples). In the last couple of years POP has undergone a resurgence with the appearance of two new planners: RePOP [Nguyen and Kambhampati, 2001] and VHPOP [Younes and Simmons, 2003] both of which present novel heuristics for POP planning and in particular for flaw selection.

### 2.3.3 Graphplan Planning

Graphplan planning introduces a new search space: search is now performed over a structure called the planning graph [Ghallab et al., 2004]. This search space is a compromise between state-based and plan-based search. Graphplan planning techniques return a partially parallel plan but it is not as unconstrained as a POP plan. The search technique allows actions to take place in parallel at discrete time steps. Planning graphs have some useful properties: planning graphs grow in polynomial size (rather than exponential state-base search) and can be constructed in polynomial time.

There are two main steps to the graphplan algorithm:

1. Construct a planning graph.
2. Search the planning graph.

A planning graph is a directed layered graph consisting of two different types of layers: propositional and action. In the very first layer are the propositions from $s_0$. In the next layer are all the actions whose preconditions exist in $s_0$. An action is not added if all its preconditions are mutually exclusive. This property is explained later. A single propositional layer and action layer make up a level. The next proposition layer is made by unioning the preceding propositional level with all the effects of all actions in the action layer.

The nodes are connected using three different kinds of edges. Between a proposition layer and action layer exist precondition edges. These record which preconditions an action in the current action layer rely on in the previous propositional layer. Between the action level and the next proposition level are the add-edges and the delete-edges. A final caveat is that no-op actions are allowed. The no-ops are used to support a single proposition between levels thus ensuring every proposition in level $l$ also appears in $l+1$. These edges function as add edges.

During graph construction it is necessary to record the mutual exclusion relations (mutexes for short) across the span of the graph. This is to make the search of a valid plan easier. There are generally two types of mutual exclusivity:

Action exclusivity is defined by interference:

- An action $a$ deletes a precondition of another action $b$
- An action $a$ deletes an add-effect of another action $b$
- The converse is also true

Proposition exclusivity can be calculated after the first set of action mutexes have been calculated:

- Two propositions $p$ and $q$ in any propositional level are mutex if every action in the same level as $p$ and $q$ that has $p$ as a positive effect is mutex with every action that produces $q$.

If actions are defined as mutually exclusive this has a knock on effect of making propositions mutually exclusive. This process is repeated as an action can also be marked mutually exclusive if a precondition of action $a$ is mutex to a precondition of action $b$. The mutex relations are propagated within the planning graph as they will be an aid to search when it comes to finding a successful plan.

The algorithm works by extending the planning graph as described above maintaining mutex relations as it proceeds. The expansion continues until all propositions in $s_g$ are present in a propositional layer with no pairs of propositions from $s_g$ mutex to each other. At this stage the algorithm attempts to find a set of actions that satisfy the goal propositions that are not mutex to each other. Once a set has been found the preconditions of the set of actions must be satisfied in the same way. If no set of actions exists the algorithm backtracks to find another set of actions. If a plan cannot be found, further expansions to the planning graph are made as long as the goal propositions appear without mutex relations before the fix-point is reached. The fix-point occurs when the algorithm produces a propositional layer that is identical to a previous layer. If the fix-point appears before the goal propositions appear without mutex relations then the problem is unsolvable.

#### 2.3.3.1 GraphPlan

GraphPlan was the first planner to solve the problem of STRIPS planning using a planning graph [Blum and Furst, 1997]. This provided a new area of research to solve the planning problem. The research was well motivated as GraphPlan outperformed the then current POP planners as GraphPlan demonstrated better scaling behaviour on the classical planning domains.

One of the main strengths of GraphPlan is that it rekindled interest in domain-independent planning. Before GraphPlan, researchers were struggling to deal with the rich structure of POP. One of Graphplan's other big strengths is that the planning graph created a source to find many useful heuristics to aid planning [Nguyen and Kambhampati, 2000]. A weakness of the early planning graph planners was inefficient memory usage, though

this has been improved by a number of authors [Smith and Weld, 1998, Long and Fox, 1999]. A number of extensions have been made to Graphplan planning so that the approach can handle more complex languages: ADL [Koehler et al., 1997], time [Long and Fox, 2003b], and metric domains [Garrido and Long, 2004].

### 2.3.4 SAT Planning

SAT Planning is an approach based on transforming the planning problem into the boolean satisfiability (SAT) problem [Ghallab et al., 2004]. The motivation is that SAT is a well studied problem for which there are many good solvers which can now be applied to planning problems once they are converted into SAT problems. A standard planning problem can be converted into a propositional formula. If a model exists for the formula, this means that a plan exists to solve the problem and that the plan can be extracted using the truth assignment.

There are a number of steps to take to transform a problem expressed in propositional STRIPS into a conjunctive normal form (CNF) formula that can then be solved by a SAT solver. Firstly the problem is restricted to finding a plan of length $n$. If there are too many actions then it is simple enough to add "no-op" actions that do nothing. Secondly, SAT planning requires the definition of a new type of proposition. A *fluent* is like a proposition except that it has a time step associated with it. This is done by adding an additional parameter to every predicate, e.g. `at(briefcase1, location1)` now becomes `at(briefcase1, location1, i)` where $i$ is the time step of the predicate.

The steps to transform a planning problem into SAT are:

1. The initial state is encoded as the conjunction of all fluents in the initial state, in addition, if a fluent $f$ is not present in $s_0$ then add $\neg f$ to the initial state.

2. The goal state is the conjunction of all fluents in the goal state at time step $n$.

3. Actions imply both their preconditions and their effects, and at all possible time steps between 0 and $n$.

4. Add additional frame axioms that state what fluents an action does not affect which deals with the "frame" problem. These simply state that if a fluent has changed value between time steps then one action that changes its value must have been executed. This is needed to prevent impossible worlds being examined.

5. Finally state that only one action can occur at time *i*.

The conjunction of the resulting logical formalisms is then converted into conjunctive normal form. This operation can be done in linear time and the resulting formula can be solved by any kind of SAT solver.

#### 2.3.4.1 SATPLAN

SATPLAN is a planner based on transforming propositional planning tasks into boolean satisfiability problems (SAT) [Kautz and Selman, 1992]. In this particular instance the authors used GSAT [Selman et al., 1992] to solve the resulting SAT problem. GSAT is categorised as a depth limited greedy hill-climber with random restarts, and it was able to outperform a standard backtracking algorithm on known hard instances. The authors then used SATPLAN to demonstrate that it can plan for a small selection of planning problems. SATPLAN was resubmitted into IPC 4 and won the award for optimal planning by using a host of new SAT solvers[1]. Hence showing its main strength of being able to take advantage of new solvers when they become available.

#### 2.3.4.2 Blackbox

Blackbox is a hybrid planning system that incorporates the planning graph into the SATPLAN framework [Kautz and Selman, 1999]. As with Graphplan, Blackbox was also an influential planner. The advantage of the hybrid approach is that the mutex information collected during the graph expansion phase is incorporated into the resulting SAT problem. The SAT problem derived from the planning graph is also more compact than previous automatic attempts. Blackbox also makes use of a then newer SAT solving algorithm called WalkSAT [Selman et al., 1994]. In an earlier work, the authors examine different methods for deriving the resulting CNF formula; this also includes the planning graph [Kautz and Selman, 1996].

### 2.3.5 Heuristic Planning

Heuristics and control rules can be applied to many areas of domain independent and domain dependent planning. This section concentrates on how heuristics have been applied to improve upon state-based search [Ghallab et al., 2004].

[1] http://www.cs.washington.edu/homes/kautz/satplan/

Returning to the simple search algorithm given in Section 2.3.1, heuristics are used to make guesses about the quality of any node in the state space. So instead of choosing a node at random a heuristic allows the search to make an informed choice. A good heuristic will allow search to expand fewer nodes and get to the goal state more quickly. The better the heuristic is at the doing this the more "informative" it is.

Heuristics, in their very nature, are not perfect. Otherwise there would be no need for the field of AI Planning. Typically a heuristic makes no guarantees that the chosen best node is in fact the best or even that the node will lead to a solution. This relates back to dead-ends and local minima in the discussion of $h^+$.

Heuristics are calculated by solving a *relaxed* planning problem. The hope is that a solution to the relaxed problem will help guide search for the real problem. As heuristics become more informative, or as the relaxed problem becomes closer to the real problem, the heuristic becomes more computationally expensive to compute.

Though heuristics do not necessarily lead to the best solution there is a way in which heuristics can be guaranteed to find an optimal solution. A heuristic is admissible if its estimates are always lower than an optimal heuristic. If the heuristic is admissible and used in conjunction with $A^*$ search then the search algorithm will always produce an optimal plan.

#### 2.3.5.1 $h^+$ in more detail

A heuristic $h(s)$ provides a distance estimate of the number of actions that need to be executed from $s$ to reach $s_g$. Using the definition of a planning problem, an action applies when $Pre(a) \subseteq s_c$ and the state is updated using $s' = s_c - Del(a) + Add(a)$. $h^+$ makes estimates based on the relaxed planning problem so that state transitions now become: $Pre(a) \subseteq s_c$ then $s' = s_c + Add(a)$, that is delete lists are ignored. As has been stated before solving this problem to optimality is still **NP**-*complete*, therefore $h^+$ has to be an estimate of the distance value. One of first papers to present the $h^+$ heuristic is [Bonnet and Geffner, 1999].

The recursive formula given here is from [Ghallab et al., 2004], replacing $\delta$ with $h$, and a couple of other adjustments to fit within the notation used in this thesis.

$$h(s,p) = 0 \; if \; p \in s$$
$$h(s,p) = \infty, \; if \; \forall a \in A, p \notin Add(a)$$
$$h(s,g) = \; if \; g \subseteq s$$

$$h(s,p) = min_a \{1 + h(s, Pre(a)) | p \in Add(a)\}$$
$$h(s,g) = \Sigma_{p \in g} h(s,p)$$

The base cases state in order:

1. If atom $p$ is present in $s$ give it a score of 0 as no action needs to be taken
2. If atom $p$ cannot be made from any action then it cannot be solved
3. If all the elements of $s$ are in $g$ then stop and return 0.

The recursive case encapsulates the idea of finding the shortest way in which to achieve all $p$ from $s$. A shortcoming of this heuristic is that it views all goals as independent, as each of the individual distances are summed. This is done to get over the **NP**-hardness of finding an exact answer. If the summation function is replaced with a maximisation function the heuristic is then admissible and can be used in conjunction with search techniques to find optimal plans. This notion of ignoring delete lists provides heuristic information for many current fully automated planners.

#### 2.3.5.2 HSP

HSP is a system based on forward-chaining heuristic search for classical planning domains [Bonnet and Geffner, 2000]. HSP made its first appearance at IPC 1 where it was the only planner based on heuristic search. The main motivation, given in the paper, is that heuristic search is already used to solve big problems in other search problems like $n$-puzzle. There is a discrepancy there as most heuristics used for solving $n$-puzzle are specific to that problem and would not work for planning in general. Heuristic planners are also flexible as slight changes to the heuristic can turn them in to optimal solvers. There have been many systems that have taken on the challenge of heuristic search, the most notable being FF [Hoffmann and Nebel, 2001] the winner of IPC 2 and a highly influential planner. Heuristic search has also been extended to solve PDDL extensions involving time and resources [Haslum and Geffner, 2001, Hoffmann, 2003]. A recent extension for HSP to optimal planning was also submitted to the IPC 4 that can solve STRIPS and metric domains optimally [Haslum, 2004].

## 2.4 Chapter Summary

This chapter presented the classical planning problem, explored the different ways in which domains can be characterised, and summarised the main search techniques for classical domain independent planning. In Chapter 5, GENPLAN will be presented which attempts to solve the classical planning problem using an EA-based approach. The classification of domains will be used to better direct empirical research into the behaviour of GENPLAN. Primarily, to examine its behaviour on a range of domains and search issues. The main search algorithms are summarised, and show the range of deterministic and non-deterministic approaches that have been applied to the planning problem. The next chapter introduces the topic of Evolutionary Algorithms.

The problem of domain independent planning is an open area of research and new search techniques are welcomed in the community. This chapter has introduced the problem of fully automated planning in the STRIPS case and discussed the major additions to the language since then. The capability of planning technology has increased massively over the last twelve years and the current best planners use a mixture of techniques presented in this chapter. What has also increased is the amount in which planners are compared, thanks mostly to the planning competitions. What has not developed in line with the empirical evaluation of planners is the understanding of the interaction between search and the attributes of the planning domain on behaviour, though Hoffman has made a influential contribution to this area [Hoffman, 2003].

# Chapter 3

# An Introduction to Evolutionary Algorithms

The primary purpose of this chapter is to introduce general EA concepts and terminology. This chapter first introduces some terminology for discussing EAs, as a variety of terms are used to name the same concept, and because the terms have been borrowed and distorted slightly from Biology. The main EA algorithms are then briefly summarised. Genetic Algorithms and Genetic Programming are more closely presented as they are the two main techniques from which GENPLAN takes its inspiration. The final section describes a theory that is used explain how GAs work.

## 3.1 Basic Terminology

EA practitioners have borrowed much of their terminology from natural genetics, in order to describe their algorithms, but they often have slightly different meanings to the biological equivalent:

- **Organism:** In the EA sense, an organism is made up of a single chromosome.
- **Chromosome or Genotype or Candidate:** A solution to a problem that can be represented internally in a computer. This thesis also uses the word *individual* for this meaning.
- **Gene:** A single element of a candidate, and collections of genes make up a chromosome.

- **Allele:** The value of the gene, for example with a bit string representation this could be either 0 or 1.

- **Genotype vs phenotype:** In biology, the phenotype is the expression of the genotype in accordance with the environment. However in most applications of EAs the genotype is the phenotype, though representations that relax this distinction are being researched.

- **Fitness Function or Objective Function:** measures the quality of an individual, and therefore allows for discrimination between individuals.

## 3.2 EA Family Tree

The best known search techniques based on the ideas of evolution are:

- **Evolutionary Programming (EP):** This is an optimisation technique that uses a population of individuals but only mutation is used to create new individuals (no crossover). The mutations are performed on individuals which can have a variety of representations (lists, real-valued vectors, finite state machines). Another point to make is that the mutation is typically adaptive over the representation of an individual, for example each real value in a vector would have its own mutation rate [Spears et al., 1993].

- **Evolutionary Strategies (ES):** This technique was developed simultaneously with Evolutionary Programming and Genetic Algorithms. There is also an East-West divide with Genetic Algorithms being developed under Holland in the United States and ESs being developed in Germany under Rechenberg. The latest variant, $(\mu,\lambda)-$ES, is an optimisation algorithm that operates almost exclusively over chromosomes made up of real number vectors. ES algorithms now use a population of individuals with future generations made up of entirely of children from the previous population. A population of size $\mu$, produces $\lambda$ children for the next population. ESs also make use of adaptable mutation rates at the allele level, like EP, but unlike EP recombination operators are used [Bäck et al., 1991].

- **Genetic Algorithms (GA):** First developed in the 1960s under John Holland [Holland, 1975]. Historically, there are a couple of key differences between GAs

and the evolutionary algorithms that were being developed at the same time. The first being that Holland wanted to not only solve problems but to use GAs as a tool to study adaptation. Algorithmically, GAs also make some important distinctions from the two previous techniques. He was the first to use a population of individuals and also the first to use three operators to generate children: crossover, mutation, and inversion. Finally Holland introduced the concept of the *schema* to produce the first theory on how GAs work. GAs, though initially dominated by bit-string representations, have used a variety of representations for candidates. In addition there have been many extensions to how populations are maintained and to the design of recombination and mutation operators [Mitchell, 1996, Michalewicz, 1999].

- **Genetic Programming (GP):** One key difference between GP and the approaches presented earlier is that the individuals being evolved are now simple programs. The field of GP grew out of the publication of John Koza's first book on GP [Koza, 1992]. The original implementation of the GP algorithm used LISP expressions, which can be executed by the LISP interpreter during evolution. The change of representation allows two things: firstly the opportunity to research autonomous programming, and secondly programs are very flexible and can be used to solve a variety of problems. Another difference between GP and other EAs is that GP often works over fitness cases. That is, the fitness value of a program is determined by running the individual on a number of cases, rather than the individual's fitness being calculated by a mathematical function. This makes GP suited to machine learning tasks. Koza also originally advocated not using any mutation when using GP.

## 3.3 A Basic Evolutionary Algorithm

Most EAs, at a basic level, are the same and comprise of the same functional components. The components are: representation of solutions, initialisation, population control, selection, fitness function, crossover, mutation, and termination. Also in common with most EAs is that decisions are made on a stochastic basis, for example crossover points are chosen at random and candidates are selected with a random bias towards fitness. This is not to be confused with pure random search because the search is directed by the selection process within EAs.

- **Representation:** An important step in implementing an EA is deciding how to represent candidates. Sometimes this step is prescribed to the researcher, early GAs were dominated by fixed length binary chromosomes, other times it requires ingenuity on behalf of the researcher. The representation of individuals will have an effect on how quickly individuals are simulated, and also how recombination and mutation operators are designed.

- **Initialisation:** Typically the first population of candidates is created randomly. It is possible to incorporate heuristics for creating initial populations of increased quality and to bias the population. Depending on the representation, it may be more difficult to generate initial candidates, tree-based GP is an example of this.

- **Fitness Function:** The fitness function is another crucial element of EA design. The fitness function is used to differentiate between solutions by assigning candidates a value reflecting their fitness. The fitness function must also be computationally fast as for most EAs the algorithm will spend the majority of its time assessing candidates. The fitness function will also describe the search landscape the EA has to search within.

- **Population:** A population of candidates is used to encode previous search attempts in the search space, to maintain a promising group of solutions, and to be a source from which new individuals are created to form new explorations of the search space. There are many ways to control the population update and a common method is to use generational control. This method creates a new population from the original population, which in turn replaces the old population. Each replacement marks a single generation.

- **Selection:** A key component of all EAs is natural selection. A mechanism must be created that reflects this concept in some way. That is, those individuals that have above average fitness also receive an above average number of opportunities to reproduce, and vice versa. The amount by which good individuals replace poor individuals is governed by the amount of *selection pressure*. Selection pressure is the rate at which the better individuals take over the rest of the population. A high selection pressure means that there is a high probability that the population will converge to a single candidate very rapidly. A low selection pressure may mean that search is too hindered to produce a solution. The issue of selection pressure relates back to the issue of exploitation versus exploration

in machine learning.

- **Crossover:** Not all EAs have a recombination step as seen in Section 3.2 but it is normally a component of many EA algorithms. The crossover (or recombination operator) is one way to create new individuals from the selected parents. Crossover is responsible for creating new individuals from the parents. The resulting children can be more or less fit than the parents. The crossover rate or frequency governs whether the crossover operator is used, if crossover is not applied, a single parent is cloned into the next population. The cloning operation is called *reproduction*.

- **Mutation:** In biological terms mutations are responsible for the development of new traits that may promote survival. Mutation has a place in most EAs and is sometimes used exclusively. Mutation is seen as a means of adding back information into populations that has been lost through evolution. A mutation operator is defined across two dimensions: the first being how much mutation occurs when the operator is applied, and the second being how frequently the mutation operator is applied.

- **Termination Criterion:** This part is responsible for stopping an EA run after a certain point has been reached. This prevents the EA from running forever. Some common termination criteria are: number of generations, amount of CPU time, and population convergence. Convergence occurs when all the individuals in the population are in fact the same. Termination criteria are not discussed for GAs and GP as any of the above can be used.

Algorithm 2 shows how the components are put together to form a generational algorithm. The next two subsections describe GAs and GP in more detail as GENPLAN has elements of both approaches.

### 3.3.1 Classical GA

John Holland was the first to develop GAs [Holland, 1975], from which the field of GAs has grown massively. There are many subsequent good texts available that develop and discuss GAs in more detail: [Goldberg, 1989, Mitchell, 1996, Michalewicz, 1999].

- **Representation:** Early work in the GA field prescribed the use of a fixed-length binary representation of solutions for all problems. Binary values would be used

**Algorithm 2** A Generational EA algorithm

```
1: Initialise the first population
2: Evaluate the individuals in the initial population and assign a fitness to each individual
3: while the termination criterion is unsatisfied do
4:    while the new population is not full do
5:       Select individuals from the old population using the selection method
6:       Perform either crossover or reproduction on the selected individuals
7:       Perform mutation operations on the children of the previous step
8:       Evaluate the children
9:       Insert the children into the new population
10:   end while
11: end while
12: Present the best individual as output
```

to represent real-valued numbers, that in turn would represent some real-world property. A simple example representation used to represent the MaxOnes problem is given [Ross, 1998]. Each bit represents a switch that can only be on or off. For example, a 16-bit chromosome represents 16 switches:

```
0001001101111111
```

GAs are considered a domain independent technique due to the flexibility of the representation and current work uses almost any representation for chromosomes.

- **Population:** GAs led the way in pioneering new population models. Other methods, besides generational, include: *Steady State*, and *Island Models*. In a Steady State scheme one population is used. Children produced by crossover and reproduction, together with the parents, can replace individuals in the current population. *Island Models* consist of multiple populations which allow transmission of individuals between the various populations and this is often used to help combat GAs getting trapped in local minima.

- **Initialisation:** GAs are typically initialised randomly. For a binary chromosome, each bit is either initialised to 0 or 1 with equal probability using a random number generator.

- **Fitness Function:** The fitness function is derived from the problem being solved. Continuing with the MaxOnes example, the problem is to have all bits in their on state. The fitness function is then to count the number of "on" bits in the chromosome. Another example fitness function from the travelling salesman problem would be the length of tour generated by a chromosome.

- **Selection:** The role of selection is to bias search to those individuals with a higher than average fitness. One early selection operator is *Roulette Wheel* or fitness proportionate selection. This selection method derives its name from being analogous to a roulette wheel where the size of each coloured slice represents the probability of choosing any individual. Each individual is allocated a slice that is equivalent in size to its fitness divided by the total fitness. The aim of this type of selection is to breed individuals the same number of times as is expected from their fitness.

- **Crossover:** Once selection has occurred then the individuals are recombined with each other. One way to achieve this is 1-point crossover. A crossover point is selected at random and is shared within the two parents. The crossover point divides each parent into two fragments. The first fragment of parent one is combined with the second fragment of parent two, and the first fragment of parent two is combined with the second fragment of parent one. The following is an example of 1-point crossover:

```
Parent One = 11111111000|00001
                        |
Parent Two = 11111111111|00000


Child One = 1111111100000000
Child Two = 1111111111100001
```

- **Reproduction:** Not all individuals go through crossover as some are copied into the next population directly via reproduction. This occurs with with probability $1 - p_c$, where $p_c$ is the probability of crossover.

- **Mutation:** Once crossover or reproduction have been performed the children go through a mutation step. A simple mutation for binary chromosomes is bit

flipping. For each child, the mutation runs along the entire length of the chromosome, flipping bits with probability $p_m$.

### 3.3.2 Classical GP

The first large and significant piece of work on attempting to evolve programs to solve problems was performed by John Koza [Koza, 1992]. The biggest difference between GP and GAs is the structure undergoing evolution. In GP the structure is a program, typically in LISP, and search is performed over all possible programs for a given domain and problem. Koza argues that a program is the only structure capable of solving a broad range of problems. GP is also used to solve different problems to GAs in that the problems are normally machine learning tasks. Banzaf and co-authors offer a more thorough introduction to the field of GP [Banzhaf et al., 1998].

- **Representation:** Trees form the basic representation scheme for most GP systems. This is very natural as most programming languages are parsed as syntax trees. GP trees have two main elements:
  - The *terminal set*, $\mathcal{T}$, which comprises of the set of all inputs to the GP system, and all functions which take no arguments. An example terminal set would be the truth values: 0, 1, and whatever inputs required by the problem (for logic-based problems). In terms of planning the terminal set would consist of all objects specified by the problem.
  - The *function set*, $\mathcal{F}$, which are the operators necessary to solve the problem. The set is tailored to the problem being solved. An example function set is a limited set of boolean operators: AND, OR, NOT, XOR. In terms of planning the function set would be the operators specified by the domain.

  An example tree with the following terminal set, $\mathcal{T} = \{\text{I0,I1}\}$ (standing for input zero and input one respectively) and function set $\mathcal{F} = \{AND, OR, NOT\}$ is presented in Figure 3.1. Typically trees are examined in postfix order (leftmost node first) during simulation.

  All terminals and functions must be chosen by the user and it is normally left up to the GP system to make the best use of them. The user should endeavour to reduce the function set size as this will reduce the size of the search space, and to keep the functions as simple as possible as GP is often capable of combining functions to achieve more complex behaviour.

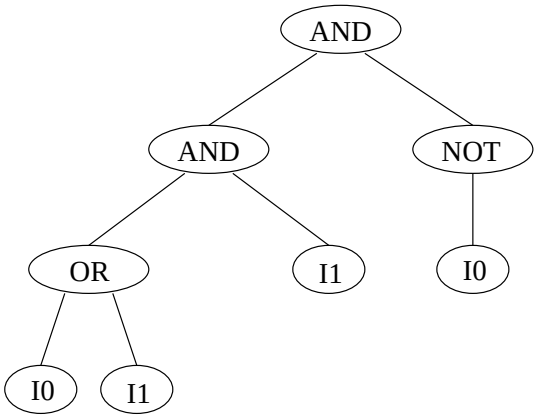



Figure 3.1: An example GP tree.

Koza's original GP systems were all written in LISP which allows the trees to be executed directly. As GP deals with programs a necessary property to satisfy is the *Closure Property*. This states that all functions must accept as input all other functions and terminals so that the programs behave gracefully.

- **Population:** GP is compatible with any previously stated population scheme.
- **Initialisation:** Initialisation for GP is harder as the shape of initial trees can affect performance of GP systems. One method for tree-based initialisation is the *grow* method. The grow method selects randomly from the non-terminal set until a prescribed depth is reached. All nodes at the next level are selected randomly from the terminal set. Other methods are described in [Koza, 1992].
- **Fitness Function:** In GP, the fitness is measured over a number of *fitness cases*. Each fitness case is made up of an input-output pairing, with the output value being the expected value for that input. The fitness function now becomes a measure of how well a program produces the desired output for each input.
- **Selection:** Any selection operator can be used in GP. A commonly used selection algorithm is called *Tournament Selection*. A subset of the population is chosen at random and the best individual from that subset is allowed to proceed to the breeding stage. The main advantages of this selection operator are that it is simple to compute, does not involve calculations of fitness for the entire population, and the user has more direct control over the amount of selection pressure in the system by varying the size of the tournament.

- **Crossover:** Tree-based crossover requires that parts of the two parent trees are swapped with each other. In standard GP crossover the point is selected randomly within each parent. The user can introduce additional bias so that functions are more likely to be selected than terminals. An example is presented in Figure 3.2, fragments to be swapped are the subtrees contained within the boxes.
- **Reproduction:** Individuals that do not undergo crossover are simply cloned into the next population.
- **Mutation:** This operator is applied to the children of either crossover or reproduction. One possible mutation operator for GP is to select a random point in a child tree, remove the existing subtree at that point, and replace it with a new randomly generated subtree. This satisfies the requirement of adding material back into the population, but could be highly damaging to an individual. There are many more possible mutations for tree-based GP [Banzhaf et al., 1998]
- **ADFs:** A unique structure to GP are Automatically Defined Functions (ADFs). ADFs are included in GP to support the idea of hierarchy in programs. Typical computer programs are made up of smaller subtasks that are repeatedly used to make a more complicated structure. An ADF consists of two parts, one branch for declaring the structure of the ADF (name, number of arguments, and the function itself), the second part being the result producing branch which is a normal GP tree. The result branch can use the declared function, like any normal function from the function set. Evolution can occur on the body of the ADF and on the result producing branch.

## 3.4 Why GAs and GPs work

The Schema Theorem underpins nearly all theoretical work into the understanding of both GAs and GPs. I will summarise the Schema Theorem for Classical GAs but the theory has been successfully transferred to GP. Holland was the first to put forward the idea of Schema and the Schema Theorem [Holland, 1975]. The Schema Theorem has been adopted and developed by others [Goldberg, 1989, Whitley, 1993]. The Schema Theorem is a widely discussed topic in the GA community for its effectiveness as a model. However recent developments have improved the usefulness of the Schema Theorem, by developing exact Schema Theories for both GAs [Stephens and Waelbroeck, 1999] and GP [Langdon and Poli, 2002]

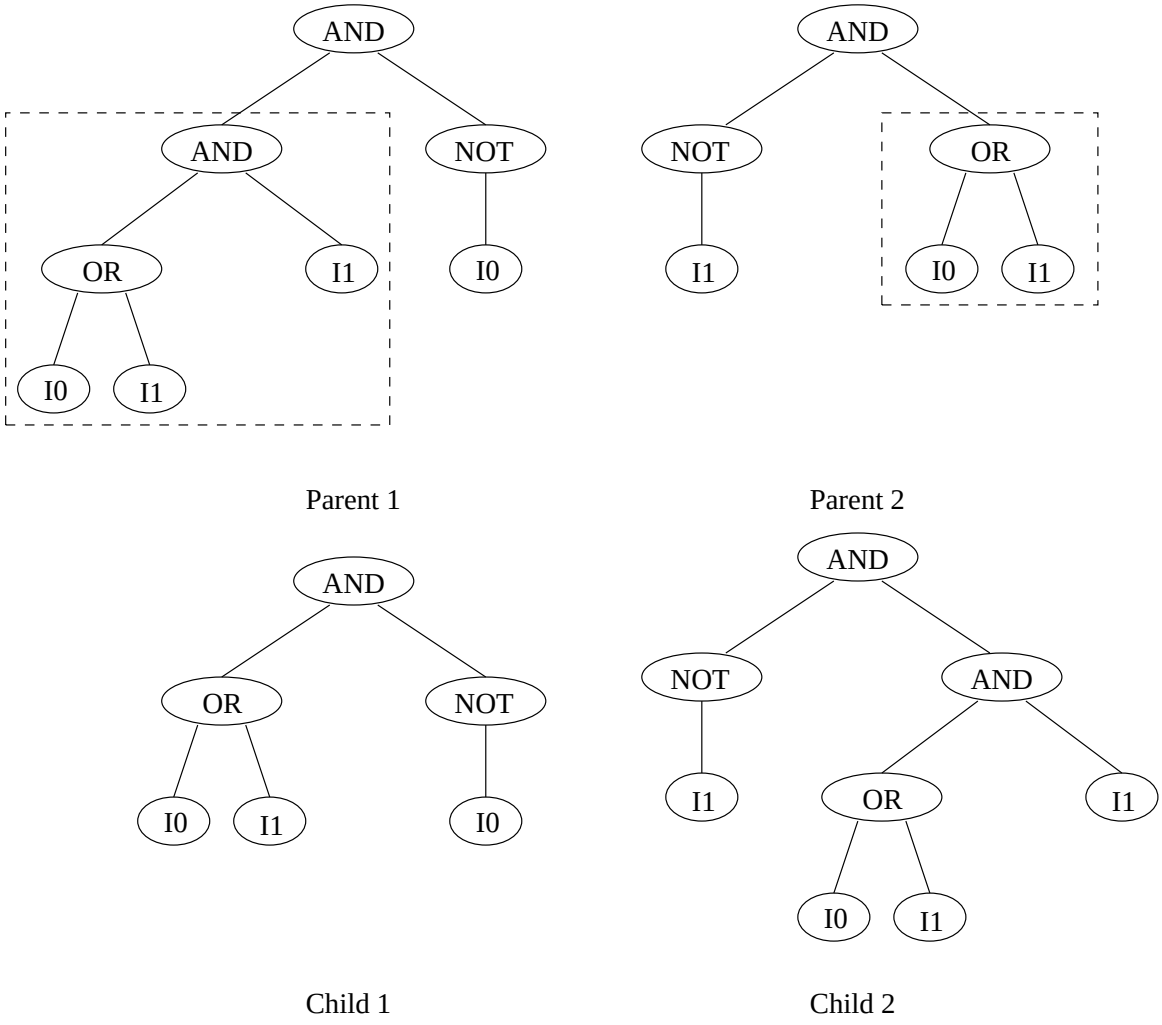



Figure 3.2: An example of standard GP crossover

### 3.4.1 Schema

Before the Schema Theorem can be discussed the theorem relies on the additional concept of the "schema." A schema is a pattern matching string for fixed length binary chromosomes. The binary alphabet is extended to include an additional symbol "*" that can match either a 0 or 1, often declared as the "don't care" symbol. $H$ is used to denote a schema and an example 7-bit schema is $H = 00*11*0$. One candidate that matches this schema is 0011100. There are in total $3^l$ schemas for a binary representation. A single individual samples $2^l$ schema, as there can only be a * or the value of the bit at each position. This shows that a single population contains much information about fitness and what could be attributing to that fitness due to the sampling.

There are two definitions relating to schema:

- $O(H)$, *order* is the number of defined bits (0 or 1), $O(00*11*0)$ is 5.
- $\mathcal{L}(H)$ *defining length* is the distance between the first and last defined bit, $\mathcal{L}(00*11*0)$ is 6.

### 3.4.2 Schema Theorem

The Schema Theorem presented here is for fixed-length binary GAs. Essentially the Schema Theorem is a pessimistic lower bound estimate of the effect of generational search on the frequency of different schemas in the population. The change in frequency of different schemas between generations is measured by examining the effect that selection, bit-flip mutation, and one-point crossover have on the number of schema. The Schema Theorem can be broken down into these component parts. The theorem is pessimistic because it assumes when a disruption event occurs due to mutation or crossover it always occurs. A disruption effect occurs when a schema is broken. This can happen with mutation if it flips a defined bit of a schema, or with crossover if a crossover point is selected within the defining length of a schema.

**The Schema Theorem**

$$E[m(H,t+1)] \geq Mp(H,t) \cdot (1-p_m)^{O(H)} \cdot \left[1 - p_c \frac{\mathcal{L}(H)}{N-1}(1-p(H,t))\right] \qquad (3.1)$$

- $E[m(H,t+1)]$ is the expected number of strings in the population, containing $H$ at generation $t+1$.

- $p(H,t)$ is the probability of selecting the schema $H$, this is usually expressed in terms of fitness proportionate selection: $p(H,t) = m(H,t)\,f(H,t)/(M\overline{f}(t))$
- $m(H,t)$ is the number of strings in the population containing schema $H$ at generation $t$.
- $f(H,t)$ is the mean fitness of the string matching H
- $\overline{f}(t)$ is the mean fitness of all the strings in the population
- $M$ is the number of chromosomes in the population
- $N$ is number of bits in the chromosome
- $p_m$ is the probability of mutation
- $p_c$ is the probability of crossover

The three elements of the Schema Theorem are:

- **Selection:** The selection term is represented by $Mp(H,t)$. The Schema Theorem given above is the "selection-free" version for the Schema Theorem as any selection function can be used to calculate $p(H,t)$. For fitness proportionate selection this term is calculated using: $p(H,t) = m(H,t)\,f(H,t)/(M\overline{f}(t))$. What this means, if mutation and crossover are ignored, schemas with above average fitness, in terms of the population average, gets more chance of being selected and placed in the next population. The converse is also true.
- **Mutation:** The mutation element is: $(1-p_m)^{O(H)}$. Using a standard bit flipping operator a schema will be disrupted if a defined bit's value changes. The above formula can be derived from standard probability.
- **Crossover:** The crossover element expresses the chance of a schema being disrupted through one-point crossover. Disruptions will occur if crossover points are selected within the defining length of a schema. This can be expressed with $\mathcal{L}(H)/N-1$. The second part of the expression specifies that a disruption event does not occur if both parents sample the same schema $(1-p(H,t))$. The complete crossover element: is $p_c\frac{\mathcal{L}(H)}{N-1}(1-p(H,t))$

The Schema Theorem represents two general notions. Short, low order schemas are less likely to be disrupted. This is expressed by both the crossover and mutation

elements of the Schema Theorem. Secondly above average schemata will increase exponentially over each generation and below average schema will decrease exponentially over each generation. This is primarily as a result of selection.

Goldberg goes further and defines the building block hypothesis [Goldberg, 1989]. The short, low-order schemata are redefined as *building blocks*. Individuals containing building blocks that result in above average fitness are preferred by the GA and the GA recombines individuals with other above average individuals. The recombination places high scoring building blocks together to create even larger building blocks and a higher fitness score. In terms of planning a plan fragment that moves one object to its goal location can be combined with another plan that moves a different object to its goal. A combined plan could successfully combine both fragments which will allow two objects to reach their goal locations. In general, we hope to build better solutions from the best current solutions, based on historical sampling of the search space.

## 3.5 Chapter Summary

This chapter has provided an introduction to the main concepts and terminology of Evolutionary Algorithms. The chapter paid particular attention to GAs and GP as these are the two branches of EAs that GENPLAN most closely resembles. Chapter 5 will explain the main differences and shared components between GENPLAN and the two general search approaches presented here. This chapter also introduces EA theory, though an adequate methodology for EA design has still yet to be born out of theoretical results. Chapter 4 summarises how EAs have been applied to planning.

# Chapter 4

# Related work in Genetic Planning

The term "Genetic Planning" is broad and encompasses three areas: planning, learning, and optimisation. Each application of EAs to one of these areas has produced initial positive results, but there has been insufficient research to determine the true extent to which EAs are suitable for each of these problems. Genetic planning is a promising area of research due to the initial positive results and lack of investigation. As the thesis deals primarily with plan synthesis this chapter will pay particular attention to other attempts at solving this task. This will done by exploring the key differences between the different approaches. The remaining two areas, learning and optimisation, are summarised for completeness and to show the breadth of genetic planning.

The four primary uses of EAs in the literature are:

1. To tackle non-standard plan synthesis tasks, those planning domains which are not experimented with by the fully automated planning community.

2. To perform classical plan synthesis, in particular using domains that are more directly relevant to the fully automated planning community.

3. To perform learning about planning, to either evolve a truly domain independent planner or to learn knowledge to supplement an existing search algorithm.

4. To optimise plans, both in terms of classical planning by reducing the number of actions in a plan, and in terms of metric optimisation by optimising plans against a user specified quality metric.

## 4.1 Genetic Planning Applied to Plan Synthesis Tasks

The main aims of the survey are to show that a single system has not been thoroughly experimented with in terms of: range of domains, different parameterisations of the system, investigating the limitations of the approach, and justifying the use of non-standard EA components. The papers in this chapter are presented in chronological order, though papers written by the same author are discussed together. The different approaches differ in certain key respects:

- Planning Domain: A careful choice of planning problem, quantity of domains, and selected problems to those domains has a major role to play in describing and understanding the performance of a system and relating that performance to the planning community as a whole. All papers discussed in this section have performed plan synthesis tasks over a very limited range of domains.

- Representation: There are a number of ways to represent plans. The approaches use a mixture of direct and indirect encodings, either structured around trees or linear representations. A direct encoding explicitly states the actions in a candidate, whereas indirect codings use an additional transformation step to determine an action. The fundamental representations used are tree-based direct encodings, linear direct encodings, and linear indirect encodings. The representations are either executed directly using procedural actions or simulated using a declarative model.

- Fitness Function: The type of fitness function used relates back to whether the planner is intended to be domain dependent or domain independent. Domain dependent fitness functions are characterised by using heuristics to augment the fitness function. Domain independent fitness functions rely on general characteristics to measure the quality of a candidate that is compatible with many domains.

- Study of Behaviour: Nearly all the papers are limited case-studies into the success or failure of using EAs to plan in a limited context. Further information about the parameterisation of the system, an examination of the relationship between search and domain, or justifications for new non-standard genetic operators are typically not given.

- Use of Specialised Operators: Some authors make use of additional genetic operators that go beyond the basic introduction to EAs given in the previous chapter. Other works define genetic operators that have already been specialised for the problem of plan synthesis. Genetic planning systems have also be modified by using ideas developed in the classical planning field.

The first paper, in the survey, applies classical GP to a robot navigation planning domain [Handley, 1994]. The domain used in this paper was a simplification of the original domain used to test STRIPS [Fikes and Nilsson, 1971]. In the domain a robot can traverse a simulated 2-D grid environment with two different rooms and a corridor connecting the rooms. The actions the robot can perform are: move, rotate, climb a block, and turn on a light. There were three different tasks for the planner to solve by directing a robot. The first task is to push two blocks up against another block, the second task is to move the robot from one location to another, and the third task is a light switching task. The third task is complicated by the fact that the light switch is raised so that it is necessary to push a block up against the wall and climb the block. This genetic planner used a tree-based representation of plans with proceduralised versions of the actions. An example plan is presented in Figure 4.1, the plan performs a knight's move. The genetic planner was able to solve every task except the light switching task. The first two tasks are relatively easy as the genetic planner is aided by the Manhattan distance heuristic [Russell and Norvig, 1995]. This is an informative fitness measure which can be assigned to an atom to reflect the number of moves required to achieve that goal atom. The third task is difficult as whether a light is on or off contains no spatial data, and so the fitness function provides no information back to the genetic planner. The problem is now akin to searching for a needle-in-a-haystack. The paper does not comment on any parameterisation experiments but does show that ADFs were useful for the first two tasks. ADFs allowed for more efficient search and resulted in more compact solutions. This paper is important as it is the first to identify a general weakness of genetic search. If the fitness function gives no information back to the algorithm, it degrades to a variant of random search, highlighting the important role the fitness function has to play.

Spector defined four classes of problem that a GP-based approach to planning may be suitable for [Spector, 1994]. The first task is directly relevant to plan synthesis and involves finding a plan to transform a single initial state to a single goal state. This paper appears later in the chapter as the three remaining tasks are machine learning tasks. The domain Spector used to test his genetic planning system was `blocksworld-2ops`.

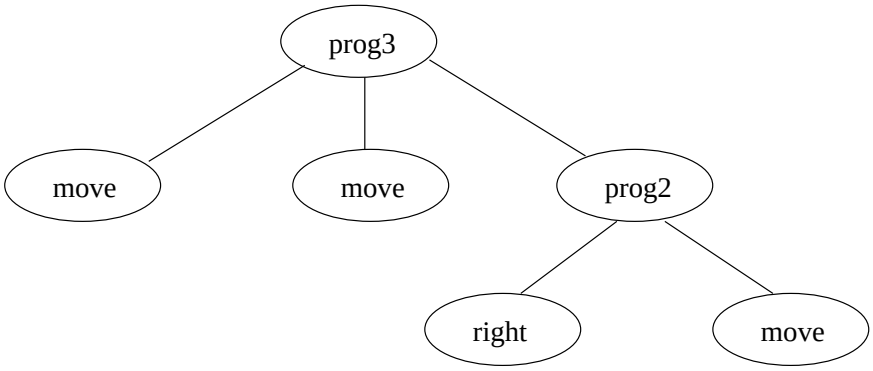



Figure 4.1: An example candidate for Handley's tree-based genetic planning system.

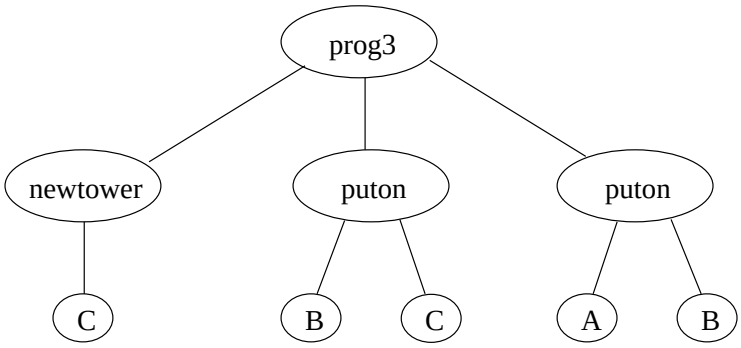



Figure 4.2: An example candidate for Spector's tree-based genetic planning system.

A specialised version of the `blocksworld` class of domains that contains two operators (`puton` and `newtower`) that rely on variables in order to execute. Spector used a classical GP system to perform planning. This entails a tree-based direct encoding for plans with proceduralised actions. An example plan is presented in Figure 4.2. The representation is similar to the previous paper except that this system has to handle arguments to actions. The system was tested only on a single 3-blocks problem for which it successfully found a plan to solve the problem. The fitness function used was the domain independent goal counting fitness function. In this paper only the population size is reported with no reporting on the behaviour of different parameterisations or the use of any specialised EA operators. However the paper does comment on the utility to use GP to find a solution and also to use the same process to find an improved solution that uses less symbols (parsimony) and less actions (plan quality).

Muslea was the first to implement a system based on classical GP that most closely resembles solving the fully automated planning problem [Muslea, 1997a, Muslea, 1997b].

He called his system "SINERGY" and used it to perform experiments on three domains: `blocksworld-2ops`, `briefcase`, and a robot navigation problem (`RNP`). In the robot navigation problem one or two robots start in an initial location on a grid and must move to a goal location. The domain is complicated by blocks which have to be pushed out of the way without obscuring a goal location. Though none of these domains are previous planning competition domains, they do closely resemble domains that have been tackled by the planning community. SINERGY uses a classical GP approach, but the representation is augmented so that it can support types. Candidates in SINERGY are equivalent to a tree, but they are "flat." An example from the `briefcase` domain is: `(take-out (put-in (t1)(t2)))`, which equates to the plan `[put-in(o1,b1),take-out(o1)]`. Most GP systems adopt the convention that each function returns its 1st argument. Muslea solves the problem of type support by enumerating all possible objects to create the terminals $t_1$ up to $t_n$. During the execution phase a terminal is cast to an object of the right type (as defined by the operator) using a simple casting function. SINERGY uses a mixture of domain dependent and domain independent fitness functions. The fitness function used for `RNP` incorporated the Manhattan distance heuristic, and for the other two domains the fitness function was based on goal counting. There is little comment on the exact implementation of the genetic operators and how parameters to the system were selected. He compared the results of SINERGY with the partial order planner UCPOP [Penberthy and Weld, 1992]. He found that SINERGY could solve larger problem instances than those that UCPOP could solve, though for the smaller instances of each domain UCPOP was more efficient.

In Ruscio's work a variable length GA was applied to the problem of creating optimal hierarchical task network (HTN) plans [Ruscio et al., 2000]. The domains and HTN rules were provided from previous research into the O-Plan planning system [Currie and Tate, 1991]. The domains are non-standard and consist of: creating a more competent adversary for an air campaign simulator, Pacifica Disaster Relief Domain[1], and a logistics-style domain. A variable length GA is much like a normal GA except that the chromosomes are allowed to grow and shrink within a maximum length. In HTN planning, basic operators can be grouped together into "schema." The schemas themselves can be placed in further hierarchies and each schema can have variables associated with it, for example what type of transport to use. Each gene in the chromosome is an integer which selects either a valid schema decomposition or a valid variable

[1] For more information on O-plan and its domains see http://www.aiai.ed.ac.uk/ oplan/

binding to a schema, as there can only be a finite set for each choice. A list of possible decompositions or bindings is made at each point of the plan when the candidate is simulated and scheduled. The integer selects one of the decompositions or bindings, after having the modulus of the number of choices applied to it. The system used a fitness function based on the makespan of a greedy scheduling of the plan derived by the chromosome, with smaller makespans being preferred. The system used standard GA operators, though different parameterisations of the algorithm are not considered. Ruscio's system had two goals: to improve upon on the simple backtracking search (used by O-Plan) in finding optimal plans and to improve upon the hand-coded scripts used for the air campaign simulator. The GA produced better solutions more quickly than O-plan for the Pacifica and logistics domains. The GA also provided better opposition for air-operations simulator. The approach taken in this paper closely resembles Grammatical Evolution [O'Neill and Ryan, 2001], a sub-field of EAs.

My own published work has made an attempt to study the effectiveness of applying EAs to classical plan synthesis [Westerberg and Levine, 2000]. This first work includes results from three domains: `blocksworld-2ops`, `briefcase`, and `logistics`. The representation used by this genetic planner is a variable length linear direct encoding with proceduralised versions of actions. Continuing with the example in Figure 4.2 a candidate is a total order list of actions: `[newtower(c), puton(b,c), puton(a,b)]`. The fitness function is domain independent, which uses the number of goals satisfied by the plan and the number of working actions contained within a plan. The genetic planner used standard genetic operators and parameter settings that were set by limited parameter experiments reported in [Westerberg, 2000]. At the time, the results were quite encouraging, on the `blocksworld_2ops` and `briefcase` problems, especially because of the nature of EA search. The `logistics` domain was unsolvable due to the relationship between the domain, fitness function, and algorithm.

I also attempted to improve the efficiency of genetic planning by improving the seeding of the initial population by using simple search [Westerberg and Levine, 2001a]. The initial candidates are built using a forward search algorithm similar to the one described in Section 2.3.1. Using simple search did produce better quality initial populations and did improve the efficiency of the system for `blocksworld-2ops`. Another attempt at improving the efficiency of the system using elite crossover is presented in [Westerberg, 2002]. The improvement was to place the parents and the children of crossover in a tournament. Only the best two individuals were placed into the next population. The results were limited but seemed to suggest an improvement in efficiency

for the `blocksworld-3ops`, `gripper`, and `miconic` domains, with no improvement for `mystery`.

Han Yu used a variable length GA to evolve plans for two planning domains: Towers of Hanoi, and N-puzzle [Yu et al., 2003]. Towers of Hanoi is the standard textbook problem of moving differently sized disks from one peg to another, given that a larger disk cannot be placed on a smaller disk. N-puzzle is the sliding tile problem of rearranging numbered tiles into the correct numerical order [Russell and Norvig, 1995]. This system uses an encoding similar to Ruscio's indirect encoding. Each gene in the chromosome is a floating point number between zero and one. During candidate simulation all applicable actions are found at the current state. One is divided by the total number of applicable actions so that each action is given a slice between zero and one. Whatever slice the floating point number from the chromosome lies in controls which action is applied to the state. As an example, if at the current state there are 10 actions which apply and the current gene value is 0.45, this means that the fourth action is selected as 0.45 lies between 0.4 and 0.5. The system also uses some non-standard components for a GA: a special crossover operator and a multi- phase GA. The system uses normal one-point crossover and an additional "state-aware" crossover operator. The second crossover operator selects a point at random in the first parent but the crossover point in the second parent must be a point where the state matches the state at the crossover point in the first parent. A multi-phase GA divides a run into several phases, for example 500 generations can be turned into 5 separate phases each consisting of 100 generations. At the end of the phase the current initial state is updated by the current best solution and search then continues with that state as the initial state but with a new random population. The search continues until a valid solution is found. The fitness functions used are not compatible with a domain independent approach. Yu's system makes use of a special function for Hanoi, and the Manhattan distance heuristic for N-puzzle. The fitness function includes an additional component to prefer shorter chromosomes. Using this approach the system was able to move 7 discs correctly for Hanoi and solve a 4x4 sliding puzzle problem. There is no discussion on how parameters were derived or justification that the additional non-standard components behave as expected, the exception being that the multi-phase approach appears beneficial. The multi-phase generational algorithm performed better than having a single large number of generations. The author hopes to extend his work to the planning and execution of complex tasks on a computational grid.

Han Yu has since improved his genetic planning system by devising a means to identify serialisable subgoals using a recursive algorithm [Yu et al., 2004]. Most planning problems can be decomposed into tasks that have an ordering. The idea is that as each sub-goal is achieved the next goal can be tackled without disrupting anything that has gone on before. The only difference between this system and the previous system is that an ordered list of subgoals are determined $[g_i, \ldots, g_n]$ using a recursive subgoal strategy based on [Korf, 1987]. The system is applied to the first subgoal $g_i$, once solved the next subgoal is attempted up until $g_n$ is solved. The plans for each subgoal are then concatenated together. This technique has improved the performance of the genetic planner as it can now solve a 7x7 sliding puzzle problem.

Brie and Morignot present a very recent addition to the field of genetic planning and is a fully automatic genetic planner which is reported in [Brie and Morignot, 2005]. The system has so far been applied to four domains: `blocksworld-4ops`, `blocksworld-2ops`, typed and untyped `gripper`. The system is very similar to GENPLAN in that it is attempting to evolve plans for typical classical planning problems by using a variable length direct encoding. However their system makes use of a complex fitness function, many mutation operators, complex termination control, and multiple populations. The aim of the improvements is to improve the efficiency of the system. They present limited parameterisation results showing the effect of differently sized populations, varying the number of populations, and varying the tournament size in tournament selection. The results are limited as the there are only two distinct domains being tested, and a total of six small problems. However they show positive results using a multi-population approach and conclude on a tournament size of three. The general tone of the paper is positive, but it lacks justification of some of the improvements made to a standard system and lacks results from a sufficient range of domains of variable difficulty and problem difficulty.

### 4.1.1 Plan Synthesis Summary

Table 4.1 summarises the different approaches across the important distinguishing features: problem domain, representation, fitness function, investigation into behaviour, and specialist operators used.

| Author | Problem Domains | Representation | Fitness Function | Behaviour Study | Special Operators |
|---|---|---|---|---|---|
| Handley | robot nav. | tree-based<br>direct encoding<br>procedural | domain dependent | initial study of ADFs | ADFs |
| Spector | `blocksworld-2ops` | tree-based<br>direct encoding<br>procedural | domain independent | none | none |
| Muslea | `blocksworld-2ops`<br>`briefcase`<br>robot nav. problem | "flat-tree"<br>partial indirect encoding<br>procedural | both | bettered UCPOP | none |
| Ruscio | 3 HTN domains | variable length linear<br>indirect encoding<br>declarative actions | domain independent | bettered backtracking<br>improved simulator | none |
| Westerberg | `blocksworld-2ops`<br>`briefcase`<br>`logistics` | variable length linear<br>direct encoding<br>declarative actions | domain independent | initial parameter study<br>initial domain study | none |
| Han Yu | Towers of Hanoi<br>N-puzzle | variable length linear<br>indirect encoding<br>procedural | domain dependent | two domain study | multi-phase<br>crossover<br>goal orderings |
| Brie | `blocksworld`<br>`gripper` | variable length linear<br>direct encoding<br>declarative | domain independent | parameter study | fitness function<br>mutation<br>population control |

Table 4.1: A summary of EA-based plan synthesis approaches.

## 4.2 Learning about Planning

John Koza was the first author to apply classical GP to a machine learning task for planning [Koza, 1992]. The problem was to find a program that could plan for any initial configuration of lettered blocks to a single goal configuration. The block stacking problem involved making a single tower that spells out the word "universal," with each block having a single letter on its side. The problem was unusual in that it allowed some very specialised operators, perhaps in order to aid search. For instance, the operator `NN`, would supply the letter of the next needed block. Using 166 fitness cases, 18 of which were hand-encoded and the rest were randomly generated, Koza found a program that could stack any initial configuration into the desired goal stack. He achieved this using standard GP and his own set of standard parameters. The result was a program that was both correct, maximally parsimonious, and used the fewest number of moves necessary. The main purpose of the work was to show that GP could induce programs which included iteration. He was the first to suggest to search first for a correct program and to then make the program more efficient and parsimonious. In addition the work shows that it is possible to induce a general planning algorithm for a planning domain.

Spector's paper is presented again here as it deals with three machine learning task relevant to planning [Spector, 1994]:

- Find a planning program that will achieve a single goal state from any initial state.
- Find a planning program that will achieve any goal state from any initial state.
- Evolve truly domain independent planners that use functions derived to be useful from different domains in order to solve the new domain.

Spector looked at the first two tasks using the `blocksworld-2ops` domain and the last task was presented but not tackled. The system used classical GP to perform the machine learning tasks. The representation used was standard tree-based GP but the learning tasks require additional functions that allow for decision making and looping. Spector successfully derived a program that could produce a plan to solve the problem of any goal state to one initial state for the class of the 3-block problems. The system also created a program that could solve any three block problem. For tasks two and three Spector used hand-coded fitness cases. This paper represents a limited attempt to apply EAs to both plan synthesis and machine learning tasks. Spector concludes that:

> ...while GP has much to offer to AI planning research, more work must be done to determine exactly how it can be best applied.

L2Plan is a system based on GP that can either learn to build a complete planner for a planning domain or learn control rules in order to prune search [Levine and Humphreys, 2003]. The learnt planner is a policy made up of a set of production rules, where each production rule will suggest an action if the conditions in the rule are satisfied. L2Plan learns a set of production rules from its own automatically made examples for a given untyped domain. L2Plan was able to learn policies that could generate correct plans for any sized `blocksworld` problems, for any sized `briefcase` problems, and when the policies are used in conjunction with breadth first search the authors were able to find a large proportion of optimal plans as well. This work is the most successful in terms of finding general planning algorithms for arbitrary domains.

One limitation of the first L2Plan system is a lack of `:typed` PDDL support as the first system only supports `:strips` domains. Logi Pétursson looked to extend L2Plan to typed domains [Pétursson, 2003]. He experimented with typed versions of `briefcase`, `gripper`, and `logistics`, each presenting its own unique technical problems to the L2Plan system. Once the problems were resolved the author concluded that it was very difficult to find policies that could solve all problems from a domain without being augmented with search for the new domains. L2Plan could find suitable polices to support bread-first search for all three domains.

HAMLET-EVOCK is multi-strategy-based system for learning control knowledge [Aler et al., 2002]. The control knowledge is developed to assist the PRODIGY 4.0 system [Veloso et al., 1995]. HAMLET-EVOCK combines both a standard incremental learning component, called HAMLET, with an approach based on GP, called EVOCK. HAMLET is a learning system based on Explanation Based Learning (EBL), and generates control rules from examples [Minton et al., 1989]. In the multi-strategy approach the knowledge learnt from HAMLET is used to seed EVOCK. The hypothesis is that the two techniques complement one another and when combined the generated control rules will have greater applicability. For `blocksworld-4ops` PRODIGY 4.0 was able to solve 85% of selected problems within a time limit using knowledge generated by the multi-strategy approach. This compares favourably with only 58% of problems being solved using EBL alone. For `logistics` PRODIGY 4.0 was able to solve 87% of problems rather than 50% using EBL alone. Interestingly, the authors found that using a pure EVOCK found a better set of control rules for the `logistics` domain when not using the seeding approach.

Kockenderfer used GP to evolve hierarchical teleo-reactive programs for a particular style of blocks stacking problems [Kochenderfer, 2003]. Teleo-reactive programs [Nilsson, 1994] use ordered lists of production rules, each production rule having its own set of conditions, and the resulting output is either an action, a set of actions, or even calls to other programs. The teleo-programs are represented using trees and are evolved using standard GP with the fitness cases created randomly. Interestingly the system was able to find a program that solved arbitrarily sized and configured block stacking problems. One weakness is that the program was only solving multiple-stack to single-stack problems whereas normal `blocksworld` problems can involve multiple-stack goal configurations.

## 4.3 Plan Optimisation

The last application area of genetic planning is domain independent plan optimisation for both classical and metric optimisation. Plan optimisation is the problem of taking a sub-optimal plan and attempting to create an optimal plan according to some metric. In [Westerberg and Levine, 2001b] a genetic planner is extended to the task of optimising plans over the number of actions. To demonstrate the approach the genetic planner was seeded with correct plans generated from policies that were hand-coded for `blocksworld-2ops`. The policies could generate plans quickly for the problems but the produced plans are sub-optimal. The plans are then evolved using crossover and mutation, as it is simple to incorporate a plan length component into the fitness function. The system was able to improve the quality of the initial plans but not to optimality. In another paper, the work was extended to include the `briefcase` domain [Westerberg and Levine, 2001c].

I have recently extended the optimisation result to include metric optimisation for PDDL 2.1 domains [Westerberg and Levine, 2004]. Rather than simply only counting the number of actions, trade-offs can be investigated, for example a trade-off between time and fuel is common. As an example, the `briefcase` domain is extended to include distances between locations. The optimal plan is now the plan that moves the briefcase the shortest amount of distance in transporting objects to their goal locations. GENPLAN outperformed some fully automated classical planners in terms of generating higher quality plans.

## 4.4 Chapter Summary

The related genetic planning work has been presented in relation to the three applications of EAs to planning: plan synthesis, learning, and optimisation. The chapter played particular attention to related work on plan synthesis. Previous work in this area has been positive about the application of EA methods to planning. The different approaches have used a limited set of domains, a variety of representations, a mixture of domain dependent and domain independent fitness functions, and standard and non-standard genetic operators. There has also been a limited amount of reporting on the general behaviour of an EA applied to planning. All the research is still in its infancy and requires a thorough examination of the applicability of EAs to plan synthesis. The next chapter describes the implementation of GENPLAN.

# Chapter 5

# GENPLAN

GENPLAN is a test bed planning system designed to allow for an initial and thorough investigation into the effectiveness of evolutionary methods for solving fully automated planning. The goal during the development of GENPLAN was to keep the implementation as clean and as simple as possible. A basic implementation will help reveal strengths and weaknesses of the approach, and the results will also be clearer as the results (Chapters 7, 8, and 9) will not be obscured by using non-standard EA operators. Some improvements to the components are described and presented in Chapter 10. Implementational details of all the EA functional components are given, along with some additional components required to solve planning. Additionally, this chapter describes differences between this approach and with classical GAs and GP and it also highlights differences between GENPLAN and previous EA-based plan synthesis approaches.

## 5.1 GENPLAN Algorithm

The first implementation of GENPLAN was originally based on the ideas in [Koza, 1992, Banzhaf et al., 1998] and also took inspiration from GAs as well [Ross, 1998]. The second implementation of GENPLAN is presented here which uses many of the same ideas as the previous system but adopts PDDL for modelling domains. The main components of GENPLAN are: representation, initialisation, fitness function, population model, selection, crossover, mutation, and termination. This chapter also introduces new components: the PDDL parser, plan simulator, plan rewriter, and elitism. Descriptions of the most important parameters are also given. GENPLAN is roughly 14k lines long and has been in development for 2 years and is written entirely in Java. The main algorithm of GENPLAN is summarised in Algorithm 3.

**Algorithm 3** The algorithm used in GENPLAN

1: Parse the domain and problem file
2: Initialise/Seed the first population
3: Simulate and evaluate the population
4: **while** the termination criterion is unsatisfied **do**
5: Copy the best individual from the previous population into the new population (elitism)
6: **while** the new population is not full **do**
7: Select an individual or individuals in the population using tournament selection depending on the genetic operator
8: Perform either reproduction or crossover on the selected individual or individuals.
9: Perform mutation on the children of the previous step
10: Simulate and evaluate the resulting children
11: Insert the children into the new population
12: **end while**
13: Find the fitness value of the best plan and replace the old population with the new population
14: **end while**
15: Rewrite the best plan, using the plan rewriter, and present as output the resulting plan

### 5.1.1 The PDDL Parser

The first step in GENPLAN is to parse both the domain file and the problem file expressed in PDDL. This parsing step is hand-coded into GENPLAN, primarily due to a lack of automatic parsing tools for Java at the start of the PhD. Because the parser is hand-coded it supports a limited, but sufficient subset of PDDL. In addition to the parser, a simulator must be built that can simulate the operators as specified in the domain file. A benefit of a hand-coded parser is that it allows the user to add custom PDDL requirements to the language, with one example given in Chapter 10. The PDDL parser supports the following requirements:

`:strips`, this allows support for domains that follow the STRIPS style of add and deletion of facts.

`:typing`, parameters to operators in the domain can now be typed.

`:equality`, this allows for tests of equality to happen in the preconditions of an operator.

`:fluents`, numeric values can be assigned, and arithmetic operations can now occur on those values. `:fluents` also allows for comparisons, within the preconditions. In addition `:fluents` allow for user specified metrics for plans.

Despite missing a number of PDDL requirements, GENPLAN has access to many domains which vary sufficiently in terms of complexity. By supporting the PDDL standard this allows GENPLAN to be tested on a range of domains that are known to the community, thus making the results more meaningful. A negative of using non-proceduralised domains is that it no longer allows GENPLAN to be applied to those domains that are out-with the current scope of the modelling language. But the results of applying GENPLAN to any such domain would have limited interprebility.

### 5.1.2 Candidate/Plan Representation

Plans are represented as variable length linear lists of sequential instantiated atomic actions. That is, each gene is an action, and the value each gene can take is any one of the possible instantiated actions that can be created from the domain and problem (the set $A_{all}$ from Chapter 2). Each atomic action contains the name of the operator and its arguments. Each candidate is allowed to grow and shrink between zero actions

to a maximum specified length. The fitness of the plan is also stored along with the candidate. The following is an example candidate using `blocksworld-3ops` as the selected domain:

$$\begin{aligned}[Step \quad 1) \quad & move-b-to-b(c,a,b)\\ 2) \quad & move-b-to-t(c,b)\\ 3) \quad & move-t-to-b(b,c)\\ 4) \quad & move-t-to-b(a,b)]\\ & Fitness = 1.0\end{aligned}$$

In fact, GENPLAN makes one more layer of abstraction. In a preprocessing step, all possible actions are instantiated by permuting all possible objects to the operators. Now chromosomes can be made up of integers, each of which indexes a particular action from the set of all possible actions. This step reduces memory requirements, and also simplifies the creation of new individuals.

One advantage of this direct encoding is that it is simple to implement. In addition this encoding does not "hide" as much information from the search algorithm, like indirect codings, as used by Han Yu [Yu et al., 2003]. Most planning problems come with a rich structure, that is lost using a indirect encoding. For instance, the sequence `[load, move, unload]` crops up many times in planning domains, but in an indirect encoding this meaning would be lost if transfered to another chromosome by crossover. A disadvantage of using a direct encoding is that it allows for non-working actions in the make up of candidates. These actions take up space in a candidate and create additional processing during simulation. Though a thorough comparison to explore the advantages and disadvantages of all the different approaches to plan representation would be interesting it is out-with the scope of this thesis.

### 5.1.3 Population Control

GENPLAN uses a standard generational algorithm to produce new generations. An entirely new population $p'$ is made from the old population $p$ and replaces $p$ and each replacement marks a generation. The population $p'$ is created using selection, the genetic operators (crossover and reproduction), and elitism.

The two most commonly used algorithms for controlling populations are generational and steady state. The generational approach was chosen over the steady state model primarily as it is simpler to use. A steady state model requires more parameterisation experiments, in particular to deal with the replacement strategy of children

within populations for which there are many options. It is also unclear whether a steady state approach would make much difference to the overall performance of GENPLAN, and the performance may even be worse due to the increase in selection pressure a steady state model entails.

### 5.1.4 Elitism

Elitism within EAs is used to preserve some number or some percentage of the best individuals in a population between generations. These individuals are copied directly into to the next population without having any of the genetic operators applied to them. GENPLAN preserves only the best individual across populations, and if there is a tie in fitness then the first individual found with that fitness is preserved. Elitism is viewed as a "common sense" operator, though it does slightly increase the selection pressure in the algorithm.

### 5.1.5 Seeding/Initialisation

Seeding is the first functional step of any EA. This component creates the first population of candidates. This "zero" population is then subsequently used for evolution. There are many potential sources for initial candidates: random creation, using a heuristic, or even using the output of another program.

As GENPLAN is based on linear chromosomes there is no need to be concerned about producing a variety of differently shaped trees [Luke and Panait, 2001]. Though a side issue for variable length linear chromosomes would be to investigate whether it is useful to have initial candidates of varying length in the initial population. The method used in GENPLAN is to initialise all candidates to be the same length.

A simple seeding method is `Random Seeding`. In this method, candidates are initialised to be plans of length $N$. For each candidate in the population, actions are selected at random from all possible actions until a candidate contains the specified number of actions. The random strategy is simple to implement but could probably be improved upon. Results from attempts to improve the seeding process are given in Chapter 10 and also in [Westerberg and Levine, 2001a].

### 5.1.6 Simulation

The simulation component takes a candidate and then attempts to apply all the actions in the candidate. The actions are applied in sequence starting from the initial state specified in the problem. The simulation step uses the transition function as defined in Chapter 2: $s' = s - Del(a) + Add(a)$ if $Pre(a) \subseteq s_c$. If the preconditions of the action are valid the state can then be updated. Each action that has unsatisfied preconditions is simply ignored, and simulation continues with the next action in the candidate. In this way, a plan can be simulated to determine the final state of the plan. The final state is stored and is used for the fitness evaluation stage. During the simulation stage various attributes of the plan can be recorded, such as how many actions successfully updated the state. This information can then be used as additional input to the fitness function.

### 5.1.7 Fitness Function

The fitness function is a very important component to any EA as it allows search to proceed with a guide rather than just randomly. Good fitness functions are hard to design as it is difficult to know in advance how the system will behave. Often evolutionary systems will exploit some weakness in the fitness function which will lead to undesired behaviour.

Useful information can be derived during the simulation stage and from the final state of the plan. The fitness function takes the output of the simulation stage and assigns a fitness value to the candidate based on the information given to it. The simplest fitness function is to count the number of goals a plan achieves by comparing how many goals in the goal state exist in the final state after simulation of the candidate. This information can then be used to assign a numeric value to the plan as a measure of its correctness:

$$\text{goal counting fitness function} = ff_{gc} = \frac{\text{goals achieved}}{\text{total goals}}$$

The fitness is a real number ranging from 0 to 1, with 1 meaning that the plan satisfies all the goals.

The main motivation behind GENPLAN was to develop a planner capable of solving the domain independent planning problem. This excludes the use of domain dependent heuristics, like the Manhattan distance, to be included into the fitness estimate. An exception to this would be to develop a system which identifies automatically which

heuristics would prove useful to solve the current problem. Though the goal counting fitness function is simple, it does not exclude the examination of any particular goal-based planning domain. In addition any heuristic is likely to encounter the same problems as planning that this heuristic would encounter, though they may be less severe, for example $h^+$. GENPLAN is compatible with $h^+$, but the heuristic is very much best used by forward search as the ideal fitness function for GENPLAN would allow for search of goals in parallel, and be informative as well. The effect of a simple extension to the goal counting heuristic is examined in Chapter 10 and it highlights the care and examination that is required when selecting fitness functions to be used with EAs.

### 5.1.8 Tournament Selection

The selection function is used to bias which individuals are allowed to proceed to the breeding stage and any standard EA selection function can be used with GENPLAN. One effective scheme for performing selection is *Tournament Selection*. It works by selecting $n$ individuals at random from the population and the best individual in the subset is used for the breeding stage [Banzhaf et al., 1998]. Tournament Selection allows the user to have some control over the selection pressure within the algorithm, with more selection pressure being exerted with larger tournament sizes. If the tournament size is reduced to one then tournament selection becomes random selection. Tournament Selection is used because it is a standard operator in the EA community and requires less computation than fitness proportionate selection [Goldberg, 1989].

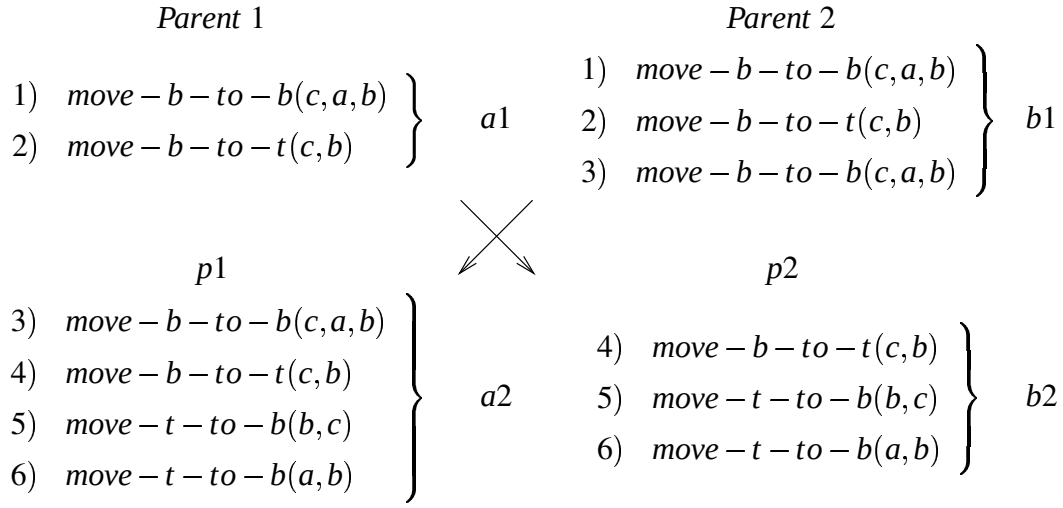
### 5.1.9 Genetic Operators

The genetic operators perform unary and binary transformations to candidates in order to produce new individuals. The resulting children are inserted into the next population. The main operator is crossover and it is a method for deciding how to swap parts of two candidates. The two other main operators are reproduction and mutation.

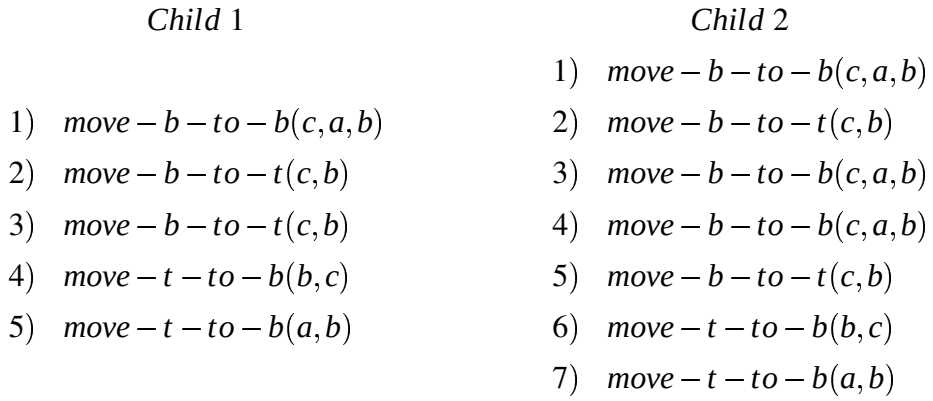
#### 5.1.9.1 Standard Crossover

Standard crossover is a variant of one-point crossover. The distinction between this and one-point crossover as defined by Poli and Langdon [Poli and Langdon, 1997] is that the two crossover points are selected at random rather than using the same crossover point for both parents.

Standard crossover takes two parents and produces two children. Standard crossover selects one randomly chosen point in each parent, $p1$ and $p2$. Child one is made up of material from the first parent up until the first crossover point, $p1$ (fragment $a1$). The rest of child one is made up material from the second parent after point, $p2$ (fragment $b2$) . Child two is made of the first fragment from the second parent (fragment $b1$) and the second fragment of the first parent (fragment $a2$).




Child one becomes the concatenation of fragment $a1$ and $b2$, child two becomes the concatenation of $a2$ and $b1$:




Once the children have been made, there are two subsequent decisions to make. The first is whether children or parents should have any subsequent operators applied to them. The second is deciding which of the parents or children should be copied into the next generation. Within GENPLAN, the children get an opportunity to be mutated and are both copied directly into the next population and the parents are discarded.

The rationale behind crossover is to allow useful transmission of information between promising parents. The ability of crossover to do this is investigated further in Chapter 9 by comparing Standard crossover with a more random crossover function.

Standard crossover achieves an undirected mixing of two parents, but in planning there is structure in plans that can be disrupted by this crossover operator. There is certainly scope for a more "intelligent" crossover operator that exploits both the representation and information collected during the simulation stage in order to direct crossover.

#### 5.1.9.2 Reproduction

Reproduction makes a perfect clone of the selected parent. The clone has an opportunity to be mutated and is then placed into the next population and the parent is discarded. Reproduction is a standard component of an EA and its purpose is to maintain promising individuals across generations.

#### 5.1.9.3 Variable Add Mutation

The mutation `VarAddMutation` adds a random number of actions, between 0 to $n$, to a candidate. Each added action is placed in a randomly selected location within the candidate. Actions are selected randomly from the set of all possible actions. The operator also works like a shift operator as all actions in front of the inserted action are shifted to right by one step. An example of the effect of this mutation operator on a candidate is shown below.

| *Parent* | *Mutated Parent* |
|---|---|
| | 1) $move-b-to-b(c,a,b)$ |
| 1) $move-b-to-b(c,a,b)$ | 2) $move-b-to-t(c,b)$ $(added\ action)$ |
| 2) $move-t-to-b(c,b)$ | 3) $move-b-to-t(c,b)$ |
| 3) $move-t-to-b(a,b)$ | 4) $move-t-to-b(a,b)$ |
| | 5) $move-b-to-b(c,a,b)$ $(added\ action)$ |

One purpose of mutation operators is to add lost information back in to the population. Therefore `VarAddMutation` is a simple instantiation of this functionality as action are added back in to candidates at random. This operator is also used as an example to show how mutations interact with the rest of the system. Chapter 7 shows that even this simple mutation can have a beneficial to search. A more promising mutation operator is investigated in Chapter 10, as it is quite unlikely that the `VarAddMutation` operator will add any needed actions. This is a simple mutation operator that satisfies the requirement of adding actions back into a population, albeit in a weak and stochastic manner.

### 5.1.10 Termination Criterion

The termination criterion prevents the EA algorithm from running forever, as EAs make no guarantees about termination. It is therefore necessary to impose an artificial stopping criterion. A common criterion, combined with a generational approach, is to terminate the algorithm after a set number of generations. Another intuitive criterion is to terminate the EA once an individual has been found that achieves a certain level of fitness. In the case of GENPLAN this would be "1" as it means that all goals have been satisfied. Finally, in the field of planning it is normal to terminate a planning algorithm after a set period of time. A combination of fitness and time makes up the termination criterion used by GENPLAN. The program either terminates after a plan has been found that satisfies all the goals or after a set period of time has passed. This termination criterion is called "Max Time and Max Fitness". One problem found using a generational counter was that it gave more time to more complex domains, giving an unfair advantage to those domains.

### 5.1.11 GENPLAN Parameters

EAs come with a host of parameters that are required to be set by the user. GENPLAN is no exception with seven parameters that are used to control the functional components of the EA algorithm. There are additional parameters which control input and output operations, like designating results files, or setting which problem and domain to be solved. The effect of differing parameter values on the behaviour of GENPLAN is given in Chapter 7. The primary parameters are:

- *Population Size:* This specifies the maximum number of individuals in the population and the number of individuals to create during the seeding stage.

- *Number of Initial Actions:* This specifics the number of actions in the initial candidates.

- *Maximum Plan Length:* This specifies the maximum number of actions a candidate can contain.

- *Tournament Size:* This controls the number of individuals that are randomly selected from the population which undergo selection.

- *Probability of Crossover:* This parameter controls the probability of whether reproduction or crossover will be used to create new individuals for the next population.

- *Mutation Frequency:* This controls what percentage of children the variable add action mutation is applied to.

- *Mutation Amount:* This controls the number of actions added to a child selected for mutation.

### 5.1.12 Plan Rewriter

The final stage of GENPLAN is the plan rewriter. Once a plan has been found that solves the problem it is often filled with redundant actions. An action is redundant if does not alter the state and the plan rewriter removes those non-working actions. The non-working actions are found by running the plan thorough the simulator. The rewritten plan is presented to the user.

## 5.2 Differences between GENPLAN and classic GA/GP

There are a number of key differences between the approach taken by GENPLAN when compared to the classical versions of EAs as presented in Chapter 3 and to the previous approaches to plan synthesis in Chapter 4:

- **Representation:** One important representational difference is that GENPLAN makes use of linear chromosomes as opposed to tree-based chromosomes. Linear representations are accepted by the GP community [Banzhaf et al., 1998]. In addition to this, there is an open question about how best to represent a plan itself. As seen in the genetic planning literature a mixture of encodings have been tried but without any clear result. The encoding used by GENPLAN seems suitable to carry out a full study for plan synthesis and is motivated by previous results [Westerberg and Levine, 2000]. In addition, the encoding differs somewhat from classical GAs as the chromosomes are allowed to change in length.

- **Simulation:** GENPLAN makes uses of an additional simulation layer rather than directly executing individuals. Using proceduralised domains means that they have to be hand-coded into the system, making it time consuming to experiment

with multiple domains and problems. An advantage of proceduralised domains is that it is easier to experiment with custom domains. On the other hand, by supporting PDDL, this gives GENPLAN access to many more planning domains. The simulation aspect is much like the process undertaken in VAL[1] to check for plan correctness.

- **Fitness Function:** In classical GP a fitness value is found through a number of fitness cases [Koza, 1992]. In normal GP the simulation stage runs each individual on each fitness case and records the output of that individual for all the fitness cases. The computed output can then be compared with the desired output to produce a fitness value. Fitness calculation for GENPLAN is different in that there is only one fitness case which is the goal state given by the problem file. This much more corresponds to the use of a fitness function within a GA. The fitness function is also very simple when compared to the one used by Brie [Brie and Morignot, 2005]. The goal counting fitness function has been used by other researchers [Spector, 1994, Muslea, 1997b, Muslea, 1997a]. A simple fitness function is to be preferred for an initial study in to behaviour and performance.

- **Genetic Operators:** GENPLAN uses simple operators in order to implement both crossover and mutation. EAs use a variety of crossover operators, in particular GAs use uniform 1-point crossover, whereas in GP it is normal to use random 1-point crossover. In the genetic planning literature, both types have been been used, with only [Brie and Morignot, 2005] stating that the uniform operator is superior to the random operator. I present some results in Chapter 9 supporting random one-point crossover, but there is little doubt there is scope for improvement in the standard crossover operator. Koza advocated the use of no mutation with GP systems, however, current GP systems are rife with mutation operators, and have a useful role to play in returning lost material. GENPLAN makes use of mutation operators. The generation model adopted by GENPLAN is a standard component.

[1] Available at: http://planning.cis.strath.ac.uk/VAL/

## 5.3 Chapter Summary

This chapter has explicitly defined all the main functional components required to develop a genetic planner. The most important components of the system are plan representation and the fitness function. GENPLAN makes use of basic EA components, because it is a testbed system designed to validate an EA approach to fully automated planning, also because it will form a useful base system to compare with, and to direct areas of further research. As GENPLAN is a stochastic algorithm, it is important to establish a good experimental methodology so that the system can be examined and this forms the next chapter.

# Chapter 6

# Statistical and Experimental Methodology

This chapter presents the statistical tools required for describing results from stochastic algorithms and the experimental methodology used to produce the results. Statistics are the tools by which researchers summarise and visualise data (descriptive statistics), and to test data in order support conclusions about the data (inferential statistics). As EAs will produce a different result for any given run the descriptive statistics play a vital role in presenting the behaviour of GENPLAN. Equally as important is to have a well documented experimental methodology as this helps promote repeatability. The experimental methodology attempts to record all the relevant decisions that were made in order to perform an experiment. It also important for the reader to understand the statistical measures and data presentations used in this thesis. Though good experimental design and statistics are important, both are difficult to achieve.

## 6.1 Basic Statistics

The basic statistics that need to be discussed for this thesis are: types of data, terminology, statistics used, and hypothesis testing in relation to the Mann-Whitney test and the non-linearity test.

### 6.1.1 Scales of Data

Data is generated from a program, in this case, GENPLAN and the data will either be used to describe the behaviour of GENPLAN or form the basis of a inference about its

behaviour. Data can come from a variety of sources and the specific metrics that are relevant to planning will be discussed in Section 6.2.2. Data is made up of samples by taking measurements of the metrics.

All statistical hypothesis tests make assumptions about the scale of data the sample must be.

- **Categorical/Nominal Scale:** In this scale data is assigned to be belonging to one group or another and it is the weakest form of data. Examples of this type of data are: true or false, or colours. These are simply the names of categories and normally a number is a associated with each category to reflect its occurrence.

- **Ordinal Scale:** The next scale of data allows for the individual data items to be ranked into smallest to largest and it also possible to have ties between data. An example of this type of data would be to rank data into different categories, for example, small, medium, and large. The data can be ordered but it is not possible to measure the difference between data.

- **Interval Scale:** The next level of data allows meaning to be attached to differences between data. Not only can we say that ten is bigger than six but that the magnitude of the difference is four. Examples of data of this type would be measured values from a thermometer. But ratios in this type of data cannot be used, for example 20 degrees centigrade is not twice as hot as 10 degrees centigrade.

- **Ratio Scale:** The final level of data allows for ratios of data items to have meaning as well differences between data. Weights are a good example of this kind of data, as an object can weigh twice as much as another. The key difference between this and the interval scale is that the zero point is no longer arbitrary.

### 6.1.2 Terminology and Notation

The terminology, descriptive statistics, and symbols used within this thesis are discussed here.

1. Population: Is the set of all possible results and is normally infinite in size.

2. Sample: A subset of the population of size $N$, where sample $X = \{x_1, ..., x_n\}$, consisting of a set of measurements of a metric of interest.

3. Independent: The outcome of one experiment does not affect the outcome of a another experiment.

4. Sample Mean: The average of all the values in the sample, and gives an idea of the behaviour of a system on average. The mean is sensitive to outliers and is calculated using: $\bar{x} = \frac{1}{N}\sum_{i=1}^{N} x_i$

5. Sample Variance: A measure of how predictable the mean is, by indicating how spread out the data are from the mean. The variance is also sensitive to outliers and is the calculated using: $s^2 = \frac{1}{N-1}\sum_{i=1}^{N}(x_i - \bar{x})^2$. The standard deviation is the square root of the variance.

6. Median: This is the "middle" value of a set of sorted data. When $N$ is odd and the sample is sorted from smallest to largest the median is the value at position $\frac{(N+1)}{2}$. When $N$ is even, the median is the average of the two middle numbers, the values at $\frac{N}{2}$ and $\frac{(N+1)}{2}$

7. Lower and Upper Quartile: There is some discussion on how best to calculate the quartiles[1]. The lower quartile is the value at position: $(\frac{1}{4})(N+1)$. The upper quartile is the value at position: $(\frac{3}{4})(N+1)$ [Mendenhall and Sincich, 1995].

8. Maximum and Minimum: Are the largest and smallest values in the sample.

9. Outliers: These are particular values that are very large or very small in relation to the main set of data and are also infrequent. Outliers must be treated with care and any strategy for removing them has to be well justified.

10. Skewness: This term describes how the mean relates to the median of a distribution. If the mean is greater than the median then distribution is "right skewed", and if the mean is less than the median then the distribution is "left skewed." This is used to describe how the sampled distribution deviates from the normal distribution.

### 6.1.3 Hypothesis Testing

Hypothesis testing allows researchers to ask "yes or no" questions about their sampled data. More specifically the question being answered is to what degree of certainty the sampled data could have arisen out of chance based on an assumption about the data.

[1] http://mathforum.org/library/drmath/view/60969.html

If the sampled data is very unlikely to have arisen by chance we can can reject the null hypothesis and draw a general conclusion about the sampled data. If the sampled data is not unlikely to have arisen by chance then we cannot make a conclusion about the data. Hypothesis testing is a principled way of drawing conclusions about data.

#### 6.1.3.1 Assumptions

Hypothesis tests make assumptions about the sampled data it is being applied to. If these assumptions are broken or approximated this can make the results of the statistical test misleading. For instance, all tests that belong to the "parametric" class of hypothesis tests assume that the sample data follows a normal distribution.

#### 6.1.3.2 Hypotheses

There are two kinds of hypothesis, $H_0$ and $H_1$. $H_0$ denotes the "null hypothesis", and $H_1$ is the "alternative hypothesis" and is normally the negation of the null hypothesis. The null hypothesis assumes that there is no significant difference between two sampled populations for a particular statistic. The alternative hypothesis specifies a difference between the two populations. If the null hypothesis is accepted this does not necessarily make it true, simply that it has not been shown to be false. If the null hypothesis is rejected, it is done to a degree of confidence, normally to 95%.

#### 6.1.3.3 Procedure

All statistical hypothesis tests have procedures that must be followed. The procedure describes any transformations that must be made to the data and any calculations that need to be made on the data in order for the test to proceed. The product of the procedure is a value that decides whether to reject the null hypothesis or not.

#### 6.1.3.4 Test Statistic

The value produced by the test statistic is used to decide the result of the hypothesis test. The test statistic returns a real value, the magnitude of which, defines how far the experimenter can go in rejecting the null hypothesis. This allows boundaries to be placed, so that any number larger or smaller than a certain value is acceptable. This defines the "critical region" which is the set of all points that allow the null hypothesis to be rejected. The critical region can occur at either extreme of the test statistic indicating a "one-sided test", or both indicating a "two-sided test." For example, in order

to reject the null hypothesis for a one-sided $z$ test at the 5% level the $z$ score, calculated by the test statistic, must be above 1.645.

#### 6.1.3.5 $\alpha$ and Decision Rules

Once the test statistic has been calculated it is possible to compare that value with the critical value. The critical value is selected by deciding upon an acceptable level of error in incorrectly asserting the alternative hypothesis called $\alpha$. Typically this value is less than or equal to 0.05, meaning there is a 5% chance of mistakingly rejecting the null hypothesis. The value of $\alpha$ is used in looking up the critical value in a table and then comparing that value with the calculated value. The decision rules specify what is to be done in each of the three possible situations. In the rules below $M$ is the median.

1. When $H_0 : M_x = M_y$ and $H_1 : M_x \neq M_y$, and if $|z| \geq z_{\alpha/2}$ then reject $H_0$ (two-sided)

2. When $H_0 : M_x \geq M_y$ and $H_1 : M_x < M_y$, and if $z \leq -z_\alpha$ then reject $H_0$ (one-sided)

3. When $H_0 : M_x \leq M_y$ and $H_1 : M_x > M_y$, and if $z \geq z_\alpha$ then reject $H_0$ (one-sided)

#### 6.1.3.6 p-values

A p-value is a calculation of how extreme a value the calculated test score actually is. Again, the researcher wants a $p$ value less than 0.05 and the calculated p-value can be be quoted as an interval or as an exact value. Reporting p-values gives more information than stating alone whether the results are significant or not. A quick demonstration of the use of p-values is given by an example. Some statisticians reject at the 5% level, others at the 1% level, but quoted p-value 0.03 allows either to occur. This is because 0.03 lies between 0.05 and 0.01 and different researchers prefer $\alpha$ to be at different levels. Quoting the p-value is more precise than stating whether a particular value is under some threshold.

### 6.1.4 Mann-Whitney test ($N > 20$)

The Mann-Whitney test [Conover, 1999] comes in many guises, it also called the Mann-Whitney-Wilcoxon test, due to the way the test was derived simultaneously and independently by different groups. The Mann-Whitney test is one member of

the set of non-parametric tests, or "distribution free" tests. These tests make no assumptions about the distribution of the population the data is being sampled from, but non-parametric tests still make assumptions about the sampled data.

The Mann-Whitney test is used to determine whether two medians come from the same population. The general idea behind this test is to combine the two samples into a single sample, sort them, and then rank the data points. A sum of ranks can then be created for each sample. If the sum is large or small then it indicates one population produces values that are larger or smaller than the other population.

#### 6.1.4.1 Assumptions

1. Both samples are random samples from their respective populations.
2. The two samples are independent and the sample points within a sample are independent of each other.
3. The distributions of the samples are equal in variance.
4. The distributions have the same shape (skewness).
5. The observed value is a continuous random variable.
6. The measurement scale within a sample is at least ordinal.
7. If the two populations differ, they only differ in respect to their median.

#### 6.1.4.2 Procedure

1. The data consists of two populations $X$ and $Y$, where $X$ consists of $n$ data points $\{x_1, x_2, ..., x_n\}$ and $Y$ consists of $m$ data points $\{y_1, y_2, ..., y_m\}$. Let $R$ denote the ranked set of the combined populations of size $N = n + m$. Finally, $R(x_i)$ and $R(y_i)$ be the rank of the values.
2. Combine the two samples into one set.
3. Rank the sample observations from smallest to largest. If there is a tie, assign the mean rank to all the items that share the same value. Record the size of each tied group.
4. Sum the ranks for Population X.
5. Calculate the test statistic.

#### 6.1.4.3 Test Statistic

The test statistic for when $n_x > 20, n_y > 20$, and $W = \sum_{i=0}^{i=n} R(x_i)$

$$z = \frac{W - \frac{n(n+m+1)}{2}}{\sqrt{\frac{nm(n+m+1)}{12} - \frac{nm(\sum_{j=1}^{g} t^3 - \sum_{j=1}^{g} t)}{12(n+m)(n+m-1)}}}$$

Where $W$ is the sum of ranks for Population $X$, $t_j$ is the size of a tied group, and $g$ is the number of tied groups. This formula uses a correction if there are ties between the two sets of data. This formula also makes use of the large sample approximation, as the sample sizes used for the thesis will be above 20. If this is the case then the central limit theorem[2] applies.

#### 6.1.4.4 Decision Rules

Either of the one-sided rules presented in Section 6.1.5.3 can be used for this thesis, as I am most interested in showing whether there has been an improvement or degradation in the efficiency of GENPLAN.

### 6.1.5 Non-linearity test

Before the non-linearity test can be discussed it is necessary to introduce three components: scatterplots, linear regression, and non-linear regression. Scatterplots are used to visually present any relationship between two sets of data. Explicitly they plot the values in two samples, X and Y, against each other. An example scatterplot is the first plot (a) which plots the timing results of two planners X and Y against each other in Figure 6.1.

The second component is linear regression. This is used to calculate a straight line of "best fit" through the data [Younger, 1979]. Linear regression is performed by estimating the coefficients for the slope and intercept of a standard line: $y = a + bx$ where $a$ is the y-intercept and $b$ is the slope. The coefficients are estimated by minimising the sum of squared errors, the "least squares," between values produced by the function and the values themselves. The second plot (b) in Figure 6.1 presents the line of best fit for the data.

[2]The central limit theorem states that the average of random samples made from any distribution will approximate to a normal distribution.

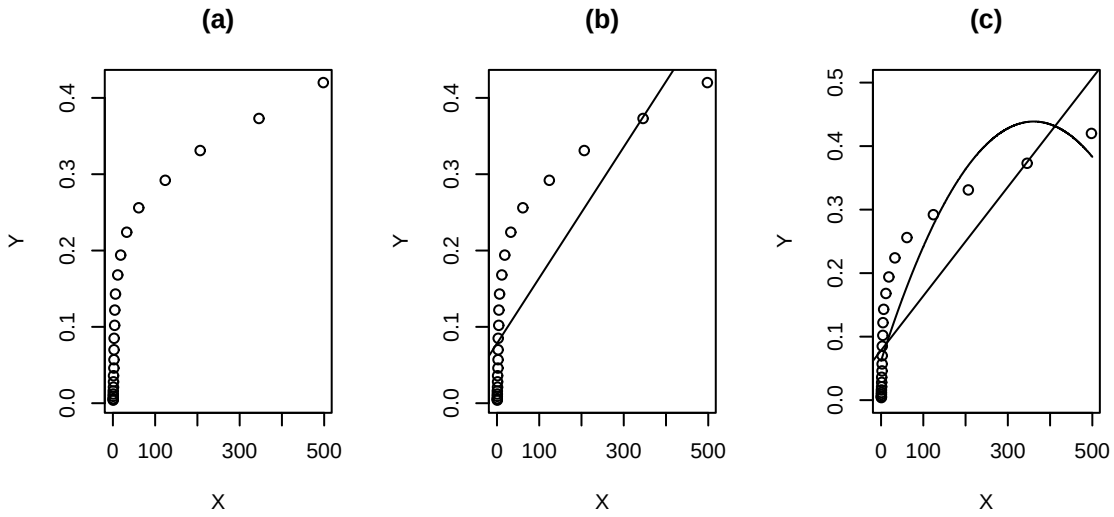



Figure 6.1: An example (a) scatterplot (b) scatterplot with linear line of best fit (c) scatterplot with quadratic line of best fit

The linear line of best fit in Figure 6.1 is not a good fit to the data and this has been confirmed visually. Another way to assess the quality of fit of a line is to calculate the $R^2$ value of the line. $R^2$ is the amount of variability of Y explained by X or put another way how well X predicts Y [Younger, 1979]. The $R^2$ value for the straight line in Figure 6.1 is 0.6945. $R^2$ is also the square of the $R$ term calculated when performing a correlation test of two variables. On the basis of the visual fit of the straight line and the $R^2$ value there appears to be a non-linearity in the data.

The third component is non-linear regression. As opposed to linear regression, non-linear regression attempts to fit any function specified by the user to the data. This can be done using the same "least squares" approach of linear regression when the functions are polynomials. I will be interested in fitting quadratic models to the data and they are an extension of the linear model: $y = a + bx + cx^2$ where $a$ is the y-intercept, $b$ is the coefficient for the linear term, and $c$ is the coefficient for the quadratic term. Estimates are again created for the three coefficients that best fit the data. The third plot (c) in Figure 6.1 presents the line of best fit for the data when the model is a quadratic. The $R^2$ value for this second curve is 0.8157, and is a higher value than before indicating a better fit.

The non-linearity test presented here is based on [Mark and Workman Jr., 2005]. In order to test for linearity it is necessary to define what it means for data to be linear.

I have paraphrased the author's definition of linearity:

> "The property of data comparing two sets of data, such that a straight line provides as good a fit (using the least-squares criterion) as any other mathematical function"

The thinking behind this test is that if functions other than linear models are fitted to the data, and that none of the other functions are a better fit, then the data is linear. It is possible to approximate any series of data points with a polynomial of high degree by using a Taylor series. However, we are more interested in fitting a function that "follows" the data rather than over-fitting the data. This criterion can be satisfied by the "least squares" method used in regression. Simple polynomials, or quadratics, will be used as the fitting function, as recommended by the authors. This is because they are simple, do not make any requirements on the collection of data, are a result of the Taylor theorem, and ultimately "our goal is not necessarily to fit the data as well as possible" [Mark and Workman Jr., 2005]. Once a quadratic has been fitted to the data it is necessary to examine the coefficients of the polynomial. A t-test [Kanji, 1999] is performed to determine whether the coefficient for the quadratic term is significant or not compared to the standard error of the coefficient. If the quadratic coefficient is significant then a non-linearity has been detected and if it is not significant then there is insufficient evidence to claim a non-linearity. To summarise the data in Figure 6.1, the $R^2$ value for the linear model is 0.6945, the $R^2$ value for the quadratic model is 0.8157, and there is a significant non-linearity as the p-value for the quadratic coefficient is 0.001280. The scatterplots, linear and quadratic regression, and t-test will all be done by the statistics package R [R Development Core Team, 2004]. The non-linear regression analysis was performed with the aid of [Crawley, 2002].

In addition, as the test will be used to demonstrate different scaling behaviours of different planners, I will be interested in the shape of the resulting curve. An example will be discussed using Figure 6.2 and Table 6.1. The first plot (a) in Figure 6.2 shows Planner Y scaling better than Planner X. This is because the curve is bent over the line of best fit. This means that Y increases at a decreasing rate to X. Also the coefficient for the quadratic term is significant. The second plot (b) in Figure 6.2 shows an inclusive result as the quadratic term is not significant. The third plot (c) in Figure 6.2 shows Planner X scaling better than Planner Y. This is because the curve is bent under the line of best fit. This means that X increases at a decreasing rate to Y. Also the coefficient for the quadratic term is significant.

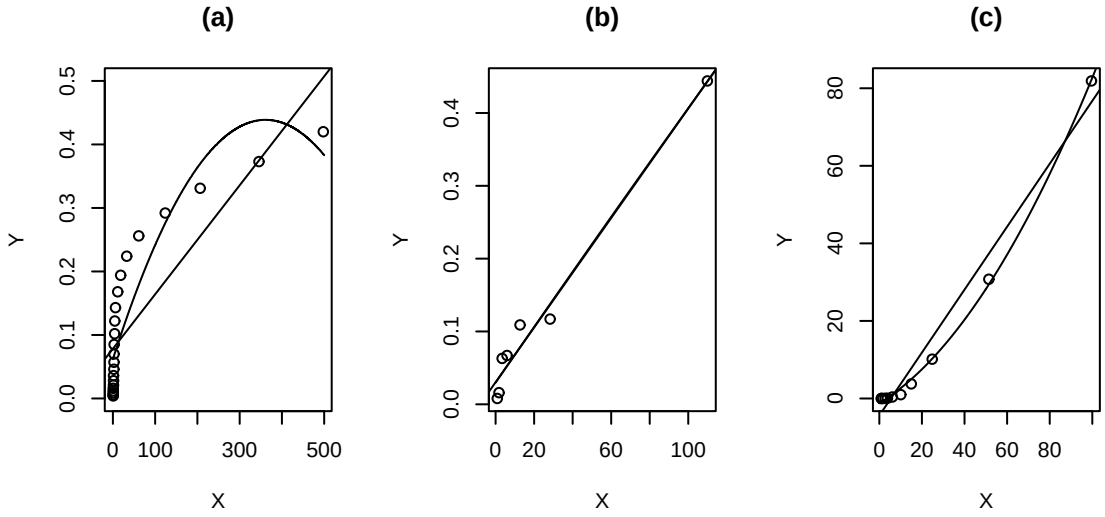



Figure 6.2: (a) Planner Y scaling better than planner X (b) An inclusive result (c) Planner X scaling better than planner Y

| Model | $R^2$ Linear | $R^2$ Quadratic | P-value |
|---|---|---|---|
| $Y \sim X$ (a) | 0.6945 | 0.8157 | 0.001280 |
| $Y \sim X$ (b) | 0.9773 | 0.9774 | 0.9637 |
| $Y \sim X$ (c) | 0.9729 | 0.9986 | $<$0.0001 |

Table 6.1: Statistical data for Figure 6.2

### 6.1.6 Problems with statistics and GENPLAN

There are a couple of problems with performing experimental work with GENPLAN. The first comes with the nature of the algorithm. Evolutionary algorithms are stochastic in nature. No two runs of GENPLAN will be exactly the same even if they share the same parameters. This requires that the algorithm is run many times on a single problem. An additional problem is that for some runs GENPLAN could fail to find a solution within the specified time. These failed runs must then be included in the descriptive statistics and be used for statistical testing, as this is the safest option. An alternative would be to create custom statistics, (e.g. the censored mean which would remove the failed values) and then perform statistical tests based on bootstrapping which then be used to perform hypothesis testing on almost any statistic [Cohen, 1995].

## 6.2 General Experimental Methodology

There is a great deal of freedom in which to perform computational experiments: selecting problems, implementation of the system, the computing environment in which the system runs, selection of performance metrics, parameterisation of the algorithm, and the reporting of results. All of the above have an effect on the overall performance of the system and the understanding of the system. Because of this freedom, it is necessary to document decisions that have been made for the experimental process in order to promote reproducibility.

The key reason for performing experiments is that AI systems, even relatively simple ones, can produce complex behaviours that cannot be predicted by simply analysing the code the system is built from. What follows are a series of quite low level experimental details and some general principles of experimentation that were adopted for this thesis. Subsequent chapters will fill in the required details when they are needed, as the text will be closer to the subject. The general methodology, with reference to [Hooker, 1995, Gent et al., 1997, Barr et al., 1995, Howe and Dahlman, 2002], followed for this thesis is:

1. Choose Goals
2. Choose Problems
3. Choose Metrics
4. Design of Experiments

5. Running the Experiment

6. Presenting the Results

### 6.2.1 Choose Goals

Experiments should not be performed on an ad-hoc basis as there should be a reason behind every experiment. In general, experiments within this thesis have a number of purposes:

1. Describe the behaviour of the system.

2. Explore the effects of different factors of GENPLAN.

3. To present the performance of the algorithm.

4. To evaluate the performance of the algorithm.

5. To a show a relationship between theory and practise.

6. Show a significant difference in performance with modified versions of the algorithm.

### 6.2.2 Choose Metrics

There are two main metrics that will be used for describing the performance of GENPLAN and they are:

- **Efficiency:** The efficiency of a planning algorithm is measured via median CPU time. The median will be calculated with the inclusion of the failure time.

- **Quality:** The plan quality is measured using the median number of working actions in the returned plan. Failed runs are not included as the returned plan will not be a valid solution.

In previous experiments performed using GENPLAN the number of generations was used to measure efficiency. This was found to be unsuitable as different parameterisations of the algorithm would vary massively the amount of CPU time GENPLAN would receive given the same number of generations. The number of generations is also not suited to comparing GENPLAN with standard planning systems.

Efficiency is instead measured using wall CPU time so that each run gets the same amount of overall processing time. Though using time as a measure has a number of deficiencies: time values are not portable to other systems and can rely on programming skill and tuning. Time is the only general measure available and is widely used in the planning community. The recorded CPU time measures the time GENPLAN was run form start to completion. The median is used over the mean, as it is often the case that when GENPLAN is applied to harder planning problems, it produces more non-normal data, in particular the data can be heavily right skewed. This makes the median a more reliable description of efficiency. It would be desirable to devise a generic metric that would aid the examination of efficiency over a range of planning systems.

Similarly the median is also used to measure plan quality. Without the inclusion of metric domains, which can have user defined quality measures, the number of actions in the plan is the standard metric for measuring plan quality for STRIPS domains. Plan quality is measured by counting the number of actions in the returned plan once all non-working actions have been stripped from the plan.

### 6.2.3 Choose Problems

This area is divided into two related tasks: choosing domains and choosing problems to those domains.

#### 6.2.3.1 Choose Domains

The selection of domains for this thesis required careful examination of many domains as there are well over 50 planning domains available[3]. The selection of domains has to be rigorous as it would be too impractical to run every possible domain and many domains are almost equivalent to each other thus making some experiments unnecessary. There has been a recent push to tackle domains of a more real-world nature [Edelkamp et al., 2004], though this is encouraging for the community, I believe these domains make it more difficult to identify problems with a new planning approach.

There are three main considerations to take into account when choosing domains. The first consideration, based on practicality, is whether the domain can by parsed and simulated correctly by GENPLAN. The second consideration is how well the domain is known, as domains from the competitions should be preferred as they are more well known by the community. A third consideration is the number and types of resource

[3] http://scom.hud.ac.uk/planet/repository/

present in a domain. The final consideration is to divide the domains into certain defining characteristics based on the search topology they create. This is important as to prevent repeated experiments with domains that behave in similar ways and to aid the analysis of the behaviour of GENPLAN.

The domains use different features of the PDDL language. The significant portions of PDDL are `:strips`, `:adl`, `:fluents`, and `:durative-actions`. Though ADL can be compiled out of domains this usually has a massively detrimental effect on the size of the search space, therefore making it impractical to experiment with those domains. Other features like `:fluents` and `:durative-actions` cannot be compiled out. Of these two, GENPLAN supports the `:fluents` requirement but not actions involving time. Additional domains that can be ruled out here are untyped versions of typed domains. Untyped domains are easy to type and typed domains have much smaller action spaces than their untyped equivalents. This leaves two main sets of domains that can be used to test GENPLAN performance. They are `:fluent` (numeric) domains and `:strips` typed domains. Typically, numeric domains include a metric in which the planner should attempt to optimise the plan against.

Resources are not excluded from the domains chosen in this chapter. Many of the domains chosen involve resources in some way, e.g limited grippers for `gripper`, multiple satellites for `satellite`, and multiple trucks and drivers for `driverlog`. Also `:fluents` can be used to better represent resources like fuel in the `mystery` domain.

The two metrics used to describe the search space of domains are minimum bench size and and maximum size of local minima. Values are derived for $ff_{gc}$ by examination of the domains and problems for those domains and by making some additional assumptions discussed below.

- *minimum bench size*

  This metric translates to the number of actions to achieve a goal. The fitness value for a candidate can only change positively when it achieves a goal. For example, from the `briefcase` domain, it takes at minimum three actions to achieve a goal (load, move, unload) because an object never starts in a briefcase or at its goal location.

  Another assumption is that the minimum bench values are quoted for the main goals of the planning problem, for example delivering a package, or taking an image, and not for trivial single action goals. This is a safe assumption because it is these goals that represent the planning problem and its these goals that cause

the most difficulty.

Chapter 2 also defined *mbed*, the maximal bench exit distance, but this metric is unsuitable for $ff_{gc}$ because the potential maximum for some planning domains is infinite. A candidate could get trapped in a loop for the entire length of a candidate never escaping a bench so absolute minimum values are used instead.

Chapter 9 will examine this issue of search in more detail.

- *maximal local minimum exit distance* (*mlmed* from Chapter 2)

  For this metric I estimate the number of actions required to reach a state where all goals become solvable without having to decrease the fitness of a plan any further in order to solve the problem. That is, it is necessary to disrupt goals which have already been achieved, lowering the fitness, so that all goals can be achieved. Continuing the example from `briefcase`, if a goal location is specified for the briefcase and the briefcase has to be moved in order to achieve other goals while at the goal location then this results in a single action local minima. As a single action is used to disrupt a perceived achieved goal so that more goals can be achieved. Chapter 9 will examine this issue of search in more detail.

Both measurements only consider positive changes in fitness, by achieving goals. It takes fewer actions in most domains to bring about a negative change in fitness by disrupting goals. Absolute minimum values and maximum values are reported here for the two metrics but the exact values can vary on a per problem basis.

There are other useful metrics as well: the kinds of dead-ends in the search space and the number of operators in the domain. Chapter 8 also makes use of the action space value as an additional measure of difficulty for the domains and problems. Complexity results for some of the domains were given in Chapter 2.

Table 6.2 summarises the main domains used for this thesis. Some specialised versions of some domains appear later in the thesis and they will be discussed when they are introduced. All the domains are reported in Appendix A and are also available online[4]. The domains that have appeared previously in a planning competition appear in bold. The number of operators defines the size of the search state space when coupled with a particular problem instance. The domains range from having none, one action local minima, to multiple one action local minima, and to `blocksworld` which

[4] http://homepages.inf.ed.ac.uk/s9638800/index.html

| Domain | Dead-ends | Number of Actions | Max Local Minima | Min Bench Size |
|---|---|---|---|---|
| Maxones | harmless | 1 | 0 | 1 |
| TTD | harmless | 2 | 1 | 1 |
| Briefcase | undirected | 3 | 1 | 3 |
| **Gripper** | undirected | 3 | 1* | 3 |
| Blocksworld-3ops | undirected | 3 | $< c$* | $< c$ |
| Numeric Mystery | unrecognised | 3 | 1 | 3 |
| **Miconic-STRIPS** | harmless | 4 | 1 | 3 |
| **Satellite** | harmless | 5 | 1* | 3 |
| **DriverLog** | undirected | 6 | 1* | 4 |

Table 6.2: A summary of the domains used in this thesis

can have variable sized and multiple local minima in a problem. The next column summarises the minimum bench size for each domain. Most goals in a problem will require an additional action, but the stated value is the absolute minimum. For some domains it is possible to have multiple local minima in the domain, this is denoted with a "*," and the letter $c$ stand for a constant value. Though more challenging domains than `driverlog` exist, these domains are not included as GENPLAN can only solve trivially small problems for `driverlog`. Also, in order to plug possible gaps in difficulty, additional domains are created, like `maxones` and `TTD`.

- `maxones`

  The goal in this domain is to switch a number of switches from "off" to "on," or from an EA perspective from 0 to 1. There is only a single operator in this domain. The operator `switch` changes the state of a switch from "off" to "on". Once a switch has been activated the switch cannot be turned off. This domain is a variant of the Travelling Salesman Problem. I felt the use of switches better reflected the intentions behind the domain, as the original TSP specification did not include distances between cities thus making the problem of finding the shortest tour non-existent. My main intention behind this domain is to make a planning domain that is as simple as possible.

- `Travelling Tourist Domain (TTD)`

  The goal of this domain is for an agent to travel between cities in order to visit

attractions at the various cities. The domain contains two operators. The first operator moves an agent between cities. The second operator is a visit operator, which allows the agent to visit a place of interest at a specific city. This domain is a variation of `buy-russell` domain [Russell and Norvig, 1995], which involves travelling to various shops and buying goods. This domain also fills a gap in difficulty between `briefcase` and `maxones`.

- `briefcase`

  This domain is a transportation domain that features a briefcase, objects, and locations. The briefcase must be used to transport objects between locations. The goal being to move the objects from their initial location to their goal location. Locations are fully connected to one another which means that the briefcase can travel to any location with a single action. Also, unlike the `gripper` domain, the briefcase can carry an unlimited number of objects. The domain contains three operators, and the first places an object in the briefcase as long as the briefcase and object are at the same location. The second removes an object from the briefcase and places the object at the same location as the briefcase. The third operator moves the briefcase between locations. This domain represents the simplest kind of transportation problem, as every location is connected to one another, and because the briefcase has infinite carrying capacity. Other domains in this chapter will impose restrictions on this transportation theme.

- `gripper`

  This domain involves a robot having to move balls between two rooms. The goal is transfer all the balls from one location to another. The domain has three operators, the first places a ball into one of the robot's grippers, the second drops a ball from a gripper into the room the robot is currently located, and the third operator moves the robot between rooms. The robot has two grippers, each able to hold a single ball. The number of rooms and grippers is fixed to two. These constraints are made because of the original implementation of the domain but there is no representational issue why there cannot be more grippers or more rooms. This domain is a restricted form of the transportation class of domains as the robot can only carry a limited number of objects. The intention behind this domain is examine the effect of a limited resource, while keeping other factors low, like number of rooms.

- `blocksworld-3ops`

  This domain is one of the variants of the `blocksworld` class of domains. This domain contains a table and blocks, which are identifiable by letters. The blocks can be in stacks of any size with one block on top of another. The goal is to take some configuration of blocks in stacks and to transform it into another configuration of stacks. This domain is sometimes called `blocksworld-noarm`, as it is missing an explicit representation of a robot arm. For this domain there are three operators. One operator moves a block from the table on to the top of another blocks. The second operators moves a block resting on another block on to the table. The third operator moves a block resting on the table to the top of another block. The original version of GENPLAN used a proceduralised two operator version of the domain [Westerberg and Levine, 2000]. Blocks is also one of the oldest planning domains and is well know within the community [Fikes and Nilsson, 1971].

- Numeric `mystery`

  This is a numeric version of the original `mystery` domain that appeared in IPC 1 which has been modified for this thesis. The goal, like `briefcase`, is to move objects between locations. The difference is that there are now resources that can be consumed. Each location has a fuel level that is decreased when the vehicle departs from that location. The fuel is non-replenishable, which makes the problem harder as there are now dead-ends in the search space. Each vehicle has a cargo capacity, in that it can only carry up to a fixed number of objects. The IPC 1 version, was a STRIPS version, that relied on a cumbersome discretized representation of fuel and cargo space. The domain has been converted to take advantage of `:fluents` requirement. This allows the resource constraints to be specific using a more intuitive numeric system. But for this instance, the metric component of numeric domains is ignored. An example operator from the original domain is given below.

```
(:operator move
  :parameters (?v - vehicle ?l1 ?l2 - location ?f1 ?f2 - fuel)
  :precondition (and (at ?v ?l1) (conn ?l1 ?l2)
                     (has-fuel ?l1 ?f1) (fuel-neighbor ?f2 ?f1))
  :effect (and (not (at ?v ?l1)) (at ?v ?l2)
```

```
                (not (has-fuel ?l1 ?f1)) (has-fuel ?l1 ?f2)))
```

The operators can be much better defined using PDDL 2.1 an example is give below:

```
(:operator move
  :parameters (?v - vehicle ?l1 ?l2 - location)
  :precondition (and (at ?v ?l1) (conn ?l1 ?l2)
                     (> (fuel-level ?l1) 0))
  :effect (and (not (at ?v ?l1)) (at ?v ?l2)
               (decrease (fuel-level ?l1) 1)))
```

Not only is the representation more intuitive it makes the domain slightly easier as the action space has been reduced, as the operators take fewer parameters. The intention behind the domain is to introduce resources into the transportation problem.

- `miconic-STRIPS`

  Miconic is a real-world inspired domain of controlling an elevator. The elevator must move people between floors and the goal of which is to move the passengers to their desired locations. Unlike the `briefcase` domain, the elevator can only move linearly between locations: there is an operator to move the elevator to a floor above the current floor and an operator to move the elevator to a floor below the current floor. There are two operators for taking on passengers and disembarking passengers. Passengers can only embark when the elevator is on their floor and only disembark when the elevator is at their goal floor. This makes the domain a little easier as is it restricts the movement of passengers.

- `satellite`

  The satellite domain is newer benchmark domain introduced in IPC 3 and takes its inspiration from scheduling satellites to record scientific data. In this domain there are $N$ satellites that have to take images of objects in the sky. There are different types of camera that take different kinds of images. The type of image is specified in the goal description. On top of this, not all types of camera are supported by each satellite. There are also a series of calibration steps that must be done before an image can be taken. It is possible to point the camera at any

of the available directions in a single step. The goal of this domain is to collect a collection of images of objects in the sky using the correct cameras. I view this domain as a more complicated version of `TTD`, as the minimum number of steps to achieve a goal has increased to 3 rather than a single visit operator. This increased minimum bench size and number of operators makes the domain much more challenging for GENPLAN.

- `driverlog`

  This is also a new benchmark domain from IPC 3. The domain involves trucks, drivers, packages, and locations. The locations are either connected by paths or roads. This produces two logical maps as a truck cannot use a path whereas a driver can traverse both paths and roads. Also, only trucks can move packages around and that is only if the truck has a driver on board. This domain is a more complicated variation of the transportation theme of planning domains. The goal is to move objects from their initial location to their goal location.

  The domain contains six operators. There are three operators for trucks and three operators for drivers. There is a move operator for moving a driver across paths and roads, and operators for disembarking and embarking a truck. Trucks can move between locations connected to each other by a road and only if a driver is in the truck. Packages can be loaded and unloaded from the truck.

#### 6.2.3.2 Problems to the domains

Problem sets to the domains will be created for different reasons and the specific choice of problems will be left on a per chapter basis. In general though, problems must be chosen in a range of difficulty from from their most simplest form up to their most challenging, for GENPLAN. The former point is determined when GENPLAN is failing to find a solution to the problem. This is necessary in order to show a range of behaviour of GENPLAN: for the domain in question, for the search issue being investigating, or for investigating the behaviour of an improvement to GENPLAN. The exact problem sets were determined by running larger sets of problems than those presented in the thesis, in order to find upper and lower bounds for problem difficulty for each domain. When problem choices are made they will be defined by the number of objects within the problem. Also for some domains it is possible to use existing problems sets created for the various competitions. In addition, some problem sets will be hand-made

in order to highlight a specific feature of search. All problem sets and random problem generators used to create problems are available online[5].

### 6.2.4 Design of Experiments

The thesis uses four different types of experiments:

- **Assessment:** This experiment is like a shot in the dark. It is the initial run of a system either using a broad range of parameters or a broad range of problems. The results of these investigations are almost never reported but they provide a singular purpose in instructing the next stage of experiments. The most useful results are to identify floor and ceiling effects.
- **Observational:** At the next level of experimentation lie observational experiments. In this instance the factors and problems have been chosen more carefully. The purpose of this type of experiment is to show relationships between factors and metrics. These experiments are used to improve the understanding of the system and to also determine the main factors that influence behaviour.
- **Manipulation:** The next level above is when there is a specific question that needs to answered. Manipulation experiments are very precise and are used to answer "yes" or "no" questions overs a single factor. For instance does improved Algorithm $A'$ perform better than the original Algorithm $A$. These judgements can be made with the aid of Hypothesis Testing.

### 6.2.5 Running the Experiment

Once factors have been to chosen to vary, problems and goals have all been chosen, it is necessary to perform the actual experiment. This subsection records the environment on which all experiments were performed and states how an experiment is run.

Luckily, Edinburgh School of Informatics runs a shared computing environment called DICE, thus making running experiments on multiple machines more consistent. The two main environmental factors are:

- Hardware
    - The computers all had 2GHz Pentium 4 processors with 512kb of cache. The machines are all single processor machines, with 512Mb of RAM.

[5] http://homepages.inf.ed.ac.uk/s9638800/index.html

- Software
    - GENPLAN is implemented in Java and compiled and run using Java 1.4.X. GENPLAN was run under under Linux 2.4.X. Small revisions were made to the Linux environment and Java SDK during experiments but these changes are unavoidable due to the shared operating environment and the number of experiments. The normal Java compiler was used using the following command: `javac *.java`. The program was run from the command line using: `time -p java -Xrunjprof -Xmx512m GenPlanV2`. GENPLAN makes use of the java virtual machine profiler in order to perform time control, requiring an additional class `runjprof`. The "-Xmx512m" parameter forces GENPLAN to work within 512Mb of memory. All experiments used the same version of GENPLAN. For the improvements chapter (Chapter 10), an attempt was to made to make changes to the code in order not to affect the behaviour of the rest of the algorithm.

A single "run" of GENPLAN includes parsing the problem and domain and then attempting to solve the problem using the assigned parameters. Once the termination criterion is reached this signals the end of a single run. Referring back to Chapter 5 the Max Time and Max Fitness criteria were used. This meant that GENPLAN is terminated either when the time limit is exceeded, or when a plan is found that achieves all the goals. For all experiments in the thesis, experimental values are calculated using 100 runs, any change in the limit is specified closer to the experiment.

### 6.2.6 Presenting the Results

The thesis uses two methods to present results from the measured values. The first primarily consists of tables. There are two versions of tables: full, Table 6.3, and half, Table 6.4. Results with full tables are available for most experiments online[6] and half versions are used to summarise the data in the thesis. The full table reports the problem name, mean with number of failures in parentheses, the standard deviation, the minimum value, lower quartile, median, upper quartile, and the maximum value. The half table reports the problem name, the mean with number of failures is parentheses, the standard deviation and the median.

The second method is the graph. Plots consist of plotting the median CPU time, or median plan length. A sample plot is given in Figure 6.3. Plotting is done using

[6] http://homepages.inf.ed.ac.uk/s9638800/index.html

| Problem | Mean (failures) | std | min | LQ | median | UQ | max |
|---|---|---|---|---|---|---|---|

Table 6.3: A full table

| Problem | Mean (failures) | std | median |
|---|---|---|---|

Table 6.4: A half table

either log scales or linear scales for the y-axis. The numbers in parentheses indicate the number of failures. For some plots it is important to examine the y-axis as the axis may not be shared across all the plots. The lines are colour coded with the problems and a legend is usually given or the plot is explained more fully in the caption.

## 6.3 Chapter Summary

This chapter has summarised the statistics used for this thesis. The chapter also presented the experimental methodology that is used to collect experimental data. The main results chapters will include more specific experimental details that pertain to that chapter. The first results chapter investigates the effect of different parameterisations of GENPLAN.

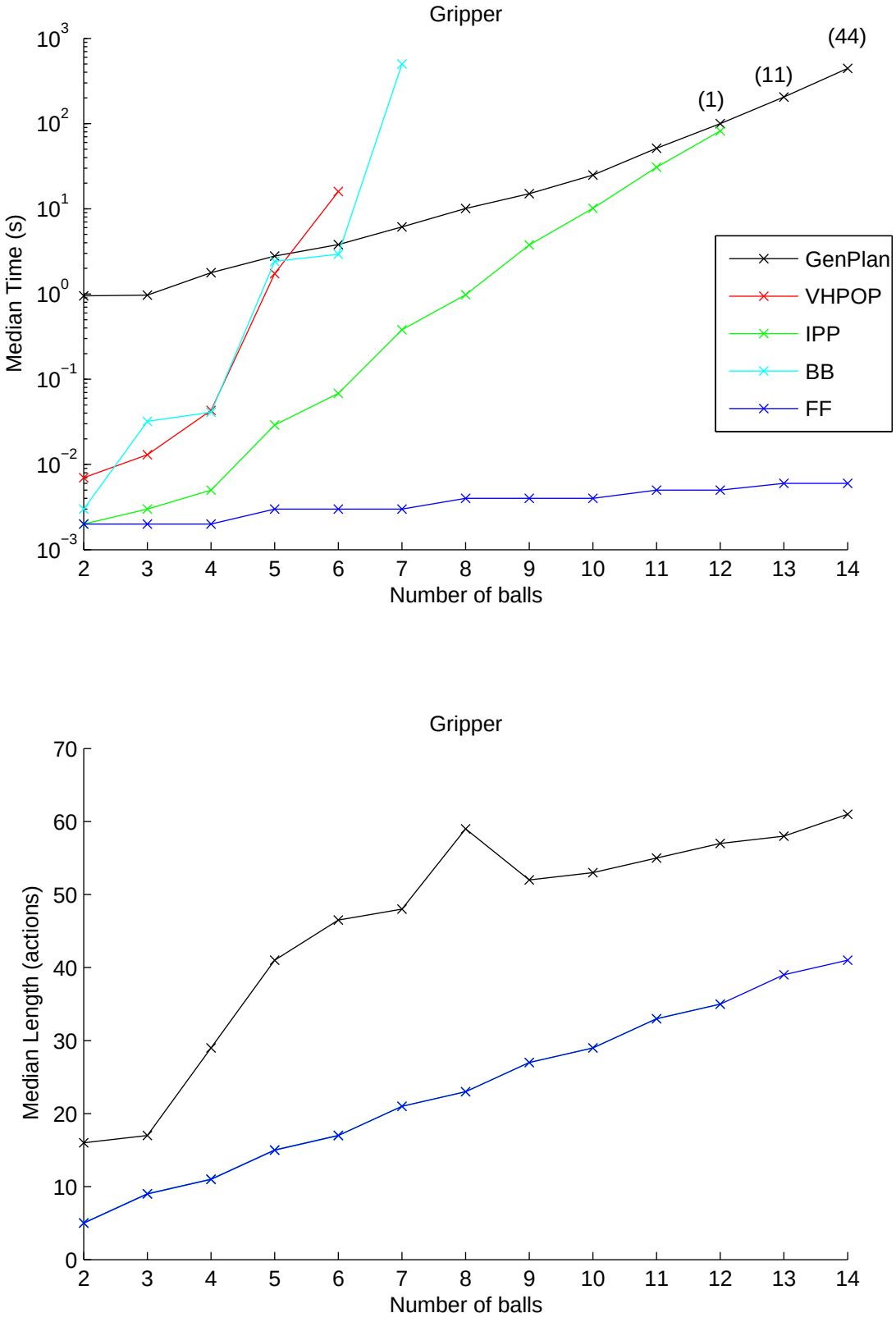



Figure 6.3: An example plot with a log y-axis plot (top) and linear y-axis (bottom)

# Chapter 7

# Parameter Exploration

Evolutionary Algorithms entail a large number of parameters which are used to configure the algorithm and to control the operation of the genetic operators. In this respect GENPLAN is no different to any other EA-based system. The first purpose of this chapter is to systematically investigate the fundamental parameters of GENPLAN. The results of this investigation will go some way to explaining the behaviour of GENPLAN. The second purpose is to use these results to create a robust parameterisation of GENPLAN that is justified and performs well across different domains and problem sizes. The parameterisation will be achieved through a series of observational experiments using a variety of settings, domains, and problems. This work is necessary for a number of reasons:

1. To better understand the behaviour of GENPLAN over a range of parameter settings, planning domains, and problems.

2. To establish a plausible setting for each fundamental parameter. A setting should allow GENPLAN to perform well over a range of domains and problems.

3. Plausible parameter settings for GENPLAN are necessary as the system will be tested against a larger corpus of domains and problems (Chapter 8).

4. The resulting settings will form a control setting which will allow for comparisons between this version of GENPLAN against any improved versions (Chapter 10).

The fundamental parameters of GENPLAN are: Population Size, Maximum Plan Length, Initial Plan Length, Tournament Size, Probability of Crossover, and mutation

(frequency and amount). Most of the parameters can have a large effect on the behaviour of the system. That is, they can radically decrease or increase the median time to finding a solution and the quality of the resulting plan. The parameters themselves can take a large range of values. This creates a very large search space of possible parameterisations to explore. There are some guidelines on parameter choice in the literature: Koza suggests settings for use with GP algorithms [Koza, 1992], and Goldberg suggests settings based on outcomes of the Schema Theorem [Goldberg, 2002]. Neither of these offer conclusive results or theories on what is best for a particular application. I thought it best to give some justification to the choice of parameters, and to study the effect of various parameter settings particularly as classical planning is an unused problem for EAs.

## 7.1 Experimental Methodology

This chapter presents results from a series of observational experiments on the fundamental parameters of GENPLAN. The experiments are observational in nature as it is not necessary to devise a parameter setting that is provably the best in all cases. This would be an impossible task as the optimum parameter setting varies on a per problem, per domain basis, and even over the course of a run. In addition, hypothesis tests are designed to answer specific questions about behaviour whereas these experiments are designed to describe the general behaviour of GENPLAN. The subsequent choice of parameter value will be chosen on the basis of trends revealed through experimentation.

Each parameter is taken in turn and GENPLAN is run with a variety of values for that parameter. The behaviour of GENPLAN over a range of parameter values is measured against a selection of domains and problems. Though the parameters are not independent, a full-scale multi-parameter test is not possible, as the number of experiments grows exponentially with the number parameters and the number of values each parameter can take. In spite of this, three two-parameter experiments were performed over a reduced problem set. Where possible, the examination of each parameter was chosen in order of importance, in order to help combat the problem of non-independent parameters. In total, the results for this chapter took roughly four months to collect, and many machine hours in the undergraduate lab. The experimental methodology set out in Chapter 6 is followed.

### 7.1.1 Choose Goals

The goal of each experiment is to decide on a particular value to each parameter as they are being considered. This will be achieved by running experiments on the chosen domains and problems over multiple values of the parameter. The differing values will reflect the behaviour of the system. In some cases multiple factors will be examined in order to reach a decision.

### 7.1.2 Choose Metrics

The efficiency of GENPLAN will be measured using median CPU time and the quality of plans will be measured using median plan length. The justifications and the means by which the metrics were recorded was presented in Chapter 6.

### 7.1.3 Choose Problems

This section is broken down into two separate subsections. The first subsection describes the choice of domains. The subsequent subsection describes the choice of the problems to those domains.

#### 7.1.3.1 Choosing Domains

Varying the parameters for a single planning domain is not sufficient. As seen in Chapter 2, planning domains can themselves be differentiated into multiple distinct subclasses. Therefore, GENPLAN was run on four different planning domains. The chosen domains for this chapter are summarised in Table 7.1. These domains represent a broad range of planning domains and encompass most known search issues: action space, local minima, and benches. The choice of domains was restricted to four due to the vast number of runs required for this chapter.

#### 7.1.3.2 Choosing Problems to the Domains

The chosen problems must produce a range of behaviours from GENPLAN, in particular the problems should range from easy to hard. Problems must not be too easy that they can be solved trivially (floor effects), and also problems must not be too hard that they are unsolvable (ceiling effects), as neither result is interesting from the perspective of choosing parameters. The set must also be kept small due to the large number of experiments that have to be run.

| Domain | Dead-ends | Number of Actions | Max Local Minima | Min Bench Size |
|---|---|---|---|---|
| Maxones | harmless | 1 | 0 | 1 |
| Briefcase | undirected | 3 | 1 | 3 |
| Blocksworld-3ops | undirected | 3 | $< c*$ | $< c$ |
| **Satellite** | harmless | 5 | 1* | 3 |

Table 7.1: The selected domains for the parameterisation experiments

| Parameter | Setting |
|---|---|
| Population Size | 1000 |
| Maximum Plan Length | 100 |
| Number of Runs | 100 |
| Maximum Time | 500s |
| Initial Actions | 20% |
| Termination Criteria | Max Time and Max Fitness |
| Tournament Size | 2 |
| Probability of Crossover | 80% |
| Add Mutation | VarAddMutation, 10%, 0-8 actions |

Table 7.2: Original Default Parameters

In order to remove some bias, all problems used in experiments were created randomly. Problems from each domain came from a mixture of either pre-existing random problem generators or from random problem generators created myself. I ran a larger set of random problems using the then default parameterisation and reduced the list down to five problems for each domain. This was done to rule out any problems that would be too hard or too easy for GENPLAN. The default parameterisation is presented in Table 7.2. These parameter settings are the result of three years worth of small scale experimentation and best guessing, but without the use of any systematic search, as presented in this chapter. In addition, it may be necessary to repeat problem selection for some domains. This is because the parameterisation of the algorithm improves over the course of the chapter and this makes some of the original test problems too easy. Again due to the massive number of runs it was necessary to restrict the number of problems to five for each domain.

- `maxones:` This is a domain where problems can be created deterministically. For each problem all $N$ switches are initialised to be off. Two problem sets were required for this domain. The problem is named by the number of switches in the problem.
  - Set 1: These problems involve 50, 54, 58, 62, and 66 switches.
  - Set 2: These problems involve 86, 90, 94, 98, and 102 switches.
- `briefcase:` I used my own random problem generator for this domain. The naming convention used for this domain is that the first number indicates the number of locations, the second the number of objects, and the last number is the number of briefcases. The goal of this domain is to transport each object from its initial location to the goal location using the briefcase. All the problems involve a single briefcase as most subsequent transportation domains presented in Chapter 8 also rely on a single carrying resource, though there are exceptions to this. Issues created by multiple resources and some types of resources are examined in Chapter 9.
  - Set 1: 3-6-1, 3-7-1, 4-6-1, 4-7-1, 5-6-1
  - Set 2: 5-7-1, 6-5-1, 6-6-1, 7-5-1, 7-6-1
- `blocksworld-3ops:` I used the random `blocksworld` problem generator BW-STATES [Slaney and Thiébaux, 2001]. The problems are all multi-stack to multi-stack problems. The first number is the number of blocks and the second is a particular instance.
  - 6-002, 7-006, 8-002, 8-003, 9-002
- `satellite:` I used the random problem generator created for IPC 3. The naming scheme is the following: the number of satellites, the number of directions (observations + targets), the maximum number of instruments per satellite, the number of modes, the number of picture goals. This makes it explicit the number of goals, and the total number of directions. The goal of this domain is to direct the satellite at the targets and take to images of the targets by correctly calibrating the satellite.
  - Set 1: 1-9-1-2-4, 1-5-1-3-2, 1-7-1-3-4, 1-7-1-3-2, 1-10-1-3-2

Some experiments require varying two parameters simultaneously. For these experiments it was necessary to report results from a single problem. The chosen problems are selected to be the second hardest problems. These problems are:

- Maxones: 62 from set 1 and 98 from set 2
- Briefcase: 4-7-1 from set 1 and 5-7-1 from set 2
- Blocksworld-3ops: 8-003
- Satellite: 1-9-1-2-4

### 7.1.4 Design of Experiments: Choosing Parameters

The order of parameter experiments was done in order of most important to least important (further justification for the order is given as each parameter is discussed):

1. Population Size
2. Population Size and Maximum Plan Length
3. Initial Plan Length
4. Tournament Size
5. Tournament Size and Probability of Crossover
6. Mutation: Mutation Amount and Mutation Frequency

The first three parameters pertain to population initialisation which is the first step of an EA algorithm. Though tournament size is a more important parameter than initial plan length, the parameters must be considered in this order so that GENPLAN can be initialised correctly.

For nearly all the parameters values chosen for the experiments were chosen from pre-parameter assessment experiments. This step is sometimes necessary in order to reduce the number of examined values of parameters as some parameters can take a large range of values. It is hoped that the reduced set will still show the full behaviour of GENPLAN. The pre-parameter experiments were performed using `blocksworld` as it has both search features of interest: a reasonable number of actions to achieve a goal, and local minima. Results from these experiments are discussed but not directly reported.

During the observational experiments all other parameters are kept constant. The original parameters are indicated in Table 7.2. Once a decision is taken for a parameter value, that value is used for the remaining experiments. For example, after the Population Size and Maximum Plan Length experiments these values are locked to their chosen values for the rest of the experiments. The implementation of GENPLAN was kept constant throughout the experiments.

### 7.1.5 Running the Experiment

The exact specification of how experiments are run was given in Chapter 6.

### 7.1.6 Presenting the Results

The results of the experiments are presented and examined using a mixture of plots and tables, as described in Chapter 6. The plots are expected to show a mixture of trends: either bowl-shaped plots with clear basins, trends for increasing or decreasing performance, or even no change in behaviour.

## 7.2 The Parametrisation Experiments

The parameters will now be examined in the order set out in Section 7.1.4.

### 7.2.1 Population Size

*Population Size* assigns the number of candidates in a population. The population size remains unchanged during a run of GENPLAN. It is widely recognised that population size is an important parameter to set correctly before using an EA [Koza, 1992, Banzhaf et al., 1998, Goldberg, 2002]. The population size is important as it controls the amount of space the algorithm has to find a solution. From the perspective of Goldberg, the population size also controls the number of building blocks in the initial population. If no new building blocks are ever created then one must ensure that all possible building blocks have a chance of being in the initial population within a minimal probability.

This first round of observational experiments considers only population sizes in order to examine the behaviour of GENPLAN across a range of sizes. Section 7.2.2 will present results from varying both population size and maximum plan length.

#### 7.2.1.1 Pre-parameter Discussion

The first step is to decide on what population sizes to experiment with, as there is a large range of values that can be assigned to this parameter. The value can range between two individuals to a population size that consumes all computational resources (several thousand). For the pre-parameterisation experiment I divided the population into three sets to determine where the interesting behaviour lies. GENPLAN was run using the following population sizes:

- **Set 1:** 10, 20, 30, 40, 50, 60, 70, 80, 90, 100
- **Set 2:** 100, 200, 300, 400, 500, 600, 700, 800, 900, 1000
- **Set 3:** 1000, 2000, 3000, 4000, 5000, 6000, 7000, 8000, 9000, 10000

The first set did not demonstrate any clear behaviours for the two metrics. The only interesting thing of note was that all five of the `blocksworld` problems were solvable using only a population size of 10 candidates and above. The second set showed a clear downwards trend for efficiency for some problems with increasing population size, but I had not yet reached a trend for degrading efficiency. The third set showed decreasing efficiency for most of the problems after a population size of 2000 candidates. This is possibly due to the overhead of maintaining larger population sizes. Plan quality was largely unaffected by the different population sizes. The final set of population sizes to be used for the first observational experiment is:

- **Final Set:** 250, 500, 1000, 2000, 4000

#### 7.2.1.2 Results

The current working intuition for population size is that it is an important factor in determining whether a problem is solvable and whether a solution to that problem will be found efficiently. That is, as problem or domain complexity goes up and as smaller populations sizes are used then GENPLAN will take longer to solve the problem and perhaps fail in solving the problem. The converse of this is considered true as well, as GENPLAN is given more material and more space then the problems should be solved more quickly and more reliably. The benefits of having ever larger and larger populations will be superseded by the cost of maintaining the population, in particular for easier problems. The efficiency of GENPLAN may improve for harder problems

when using much larger population sizes. The overhead of larger populations is derived from the amount of work required to simulate that population. The quality of the resulting plans should remain constant as population size has no direct impact on the length of plans, unlike mutation or maximum plan length.

Figure 7.1 plots median solution time against population size for each of the five chosen problems per domain.

- **Maxones:** This domain shows a population size of 500 and 1000 candidates performing well. The plot also shows degrading efficiency as the population size increases above 1000 individuals.
- **Briefcase:** This domain shows 2000 and 4000 candidates as the best performing population sizes. Also to note is the strong improvement of GENPLAN between 1000 and 2000 candidates and a slight decrease in efficiency for most problems at 4000 candidates.
- **Blocksworld-3ops:** The picture for blocks is much less clear, with either 2000 or 4000 candidates producing the best efficiency results. Most problems are being solved more inefficiently with the larger population sizes.
- **Satellite:** The plot reveals no clear trends for this domain, but a population size of 1000 candidates looks like a good choice.
- **Summary:** As can been seen from the plots, the best population size varies with both problem and domain, which presents a problem for choosing a sensible parameterisation. GENPLAN also behaves differently depending on the hardness of the domain. For easy domains like `maxones`, smaller population sizes are to be preferred. The `briefcase` and `blocksworld-3ops` domains are harder and prefer larger population sizes for some problems, but this is done at the expense of easier problems in the domain. The `satellite` domain clearly shows that ever increasing population sizes are not the answer to competent behaviour across all domains, as it is quite impervious to the changing population sizes. Perhaps some of the other parameters will alleviate the problems in this domain.

Figure 7.2 plots median plan length against population size for each of the five chosen problems per domain.

- **Maxones:** As expected, `maxones` demonstrates constant lengths across all population sizes. This is the case as any action attempting to switch on a switch that

is already on will fail and get stripped out of the final plan. For this reason the quality of plans for `maxones` will not be discussed in the remaining parameter experiments.

- **Briefcase:** This domain shows a slight upwards trend of increasing plan length as population size increases. The differences are not severe.

- **Blocksworld-3ops:** This domain shows fairly constant behaviour for plan quality over the range of population sizes.

- **Satellite:** This domain shows four problems with slight trends of increasing final plan lengths as the population size increases.

- **Summary:** These results suggest a slight preference for smaller population sizes, because of the results from the `briefcase` and `satellite` domains.

The final value for the population size will not be chosen here as another parameter, maximum plan length, needs to be considered in parallel with the population size. Together the population size and maximum plan length describe the overall size of the population.

### 7.2.2 Maximum Plan Length

The *Maximum Plan Length* parameter controls how long a candidate can get. A maximum plan length of 100 means that a candidate can have at most 100 actions in the candidate. Crossover and mutation operators are implemented so that this limit cannot be ignored. A maximum plan length parameter is necessary in order to constrain the length of candidates as they grow during a run.

Population size, together with maximum plan length, control the total space available to GENPLAN. For example, if there are 2000 candidates and a maximum plan length of 200 actions then there is space for up to a total of 400000 actions, but discretized over 2000 pieces. Together, these parameters also describe the shape of the population. For instance, in the previous example the candidates are short in comparison to the total number of candidates. This situation can be reversed by the swapping the values of the two parameters. In this case, the candidates can grow to enormous lengths but there are fewer candidates.

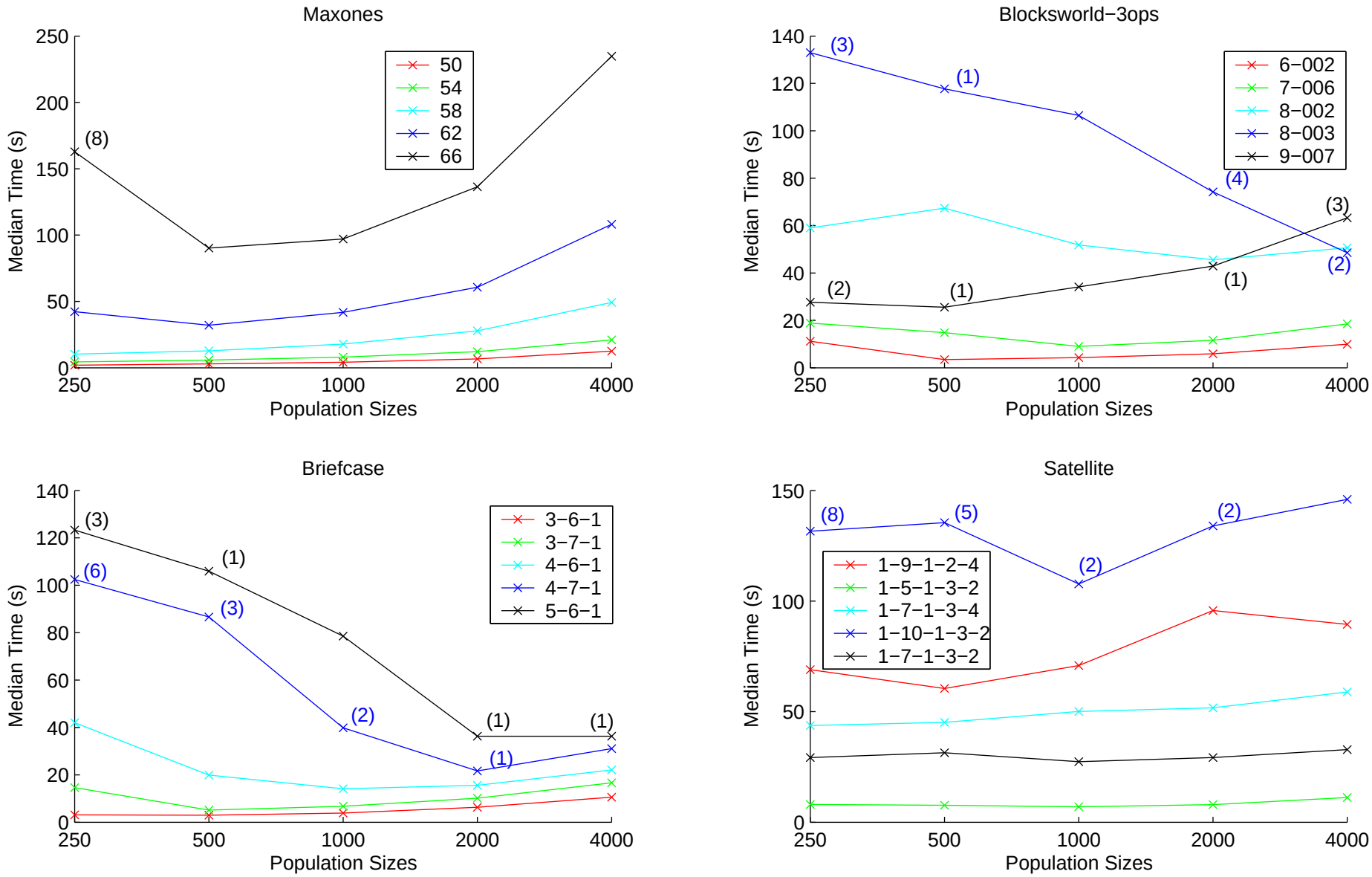



Figure 7.1: The effect of varying population size on system efficiency

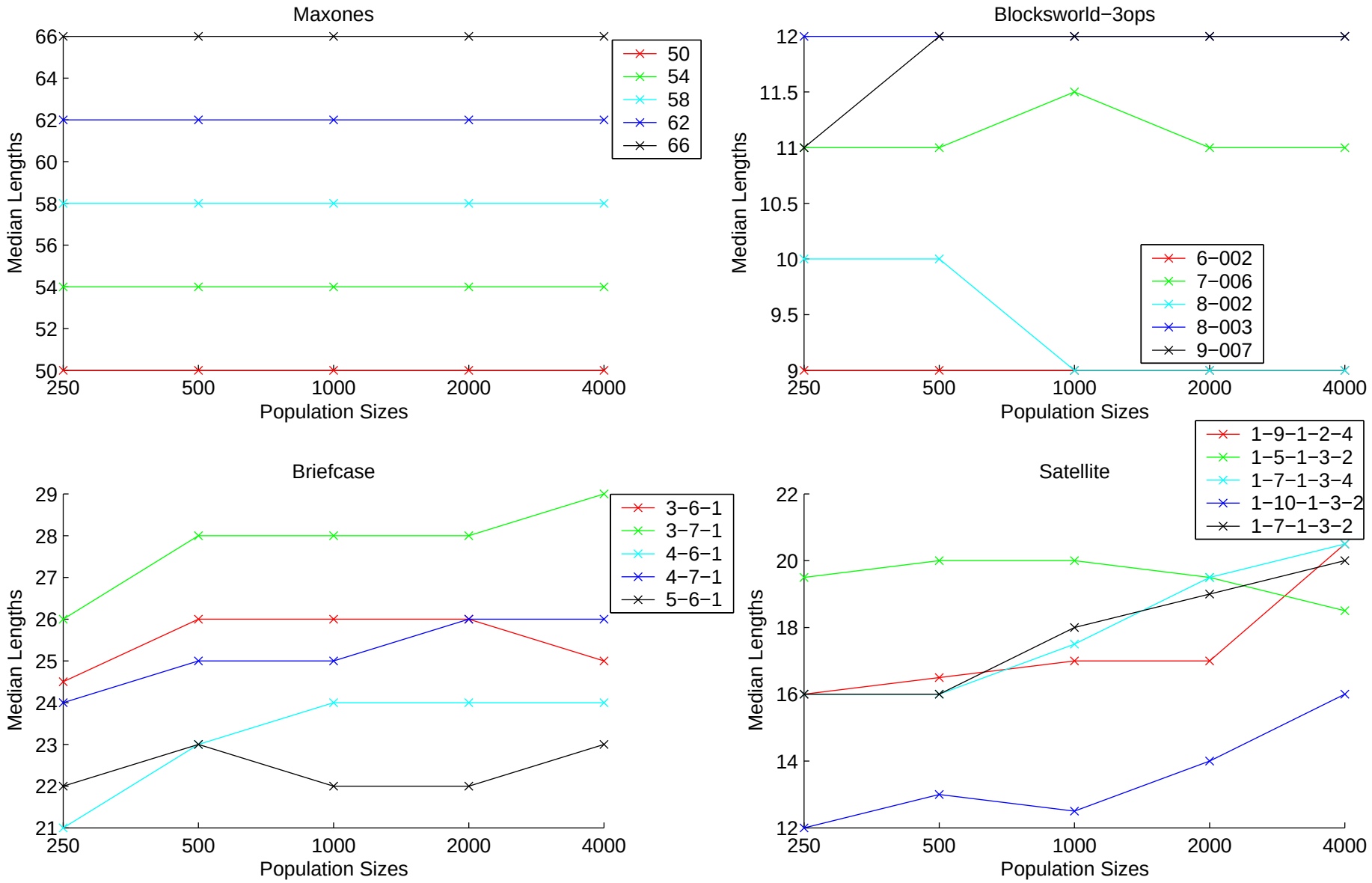



Figure 7.2: The effect of varying population size on plan quality

#### 7.2.2.1 Pre-parameter Discussion

It is necessary to reduce the set of possible values for the maximum plan length parameter. There are strong reasons for choosing an absolute minimum and maximum value for this parameter. A lower bound of 100 actions seems sensible. Any problem that has an optimal plan length greater than 100 actions would no longer be solvable. It is probably the case that such a problem is beyond the capability of GENPLAN, though a few problems from `maxones` have this property. Likewise if the maximum plan length and the population size are both set to a 1000, then this means that there is space for 1000000 actions. This equates to roughly 20Mb of memory. Therefore this places an upper bound on what the maximum plan length can be in relation to the population size.

There was no need for a pre-parameterisation experiment as the maximum plan length is already reasonably constrained due to practical reasons. The final set of parameters used for the maximum plan length experiment are fairly broad and the highest values consume large amounts of memory (roughly 280Mb). This experiment was run differently to the population size experiment. As this experiment varies two parameters rather than just one this exponentially increases the number of runs that have to be done. This forced a change in that only a single problem was run for each domain.

The selected values are listed below, the first number indicates the population size, and the numbers afterwards indicates the different maximum plan lengths used with that population size.

- Populations Sizes: Maximum Plan Lengths
- 250: 100, 200, 400, 800
- 500: 100, 200, 400, 800
- 1000: 100, 200, 400, 800
- 2000: 100, 200, 400, 800
- 4000: 100, 200, 400, 800

#### 7.2.2.2 Results

The purpose of this observational experiment is to observe whether it is more efficient to have many candidates with short maximum plan lengths than to have few candidates

with very long maximum plan lengths. The intuition here is that GENPLAN will plan more efficiently when it has more candidates in the population rather than allowing each candidate more space to grow. The candidates must have a reasonable maximum plan lengths in order to leave some space to grow in order to solve a problem. I believe it is more valuable to search to have many candidate solutions as this increases the sampling of the search space. Also, as the maximum plan length is increased, GENPLAN will return, on average, longer plans as the maximum plan the candidates have much more space to grow. In addition to this, the maximum plan length has an indirect effect on the initial plan length. Candidates are initialised to a length relative to the maximum plan length. The current default value is 20% which means a candidate with a maximum plan length of 100 is initialised to a length of 20 actions, a maximum plan length of 200 is initialised to a length of 40 actions, and so on.

Table 7.3 shows median solution time for each population size, maximum plan length pairing for each domain on a single problem. The best median times are highlighted in bold.

- **Maxones:** The best values are around the setting 250/800. This domain prefers smaller population sizes with plenty of room to grow into for each candidate.
- **Briefcase:** The best values are again around the setting 250/800.
- **Blocksworld-3ops:** The best values are around the setting 1000/400. As blocks is a harder domain the extra candidates are having a beneficial effect on efficiency.
- **Satellite:** This domain exhibits very different behaviour to other domains with a definite preference for a shorter maximum plan length and a medium amount of individuals with the best values being around the setting 500/100.
- **Summary:** Each domain presents its own preference for the proportion between population size and maximum plan length. From an efficiency perspective the harder domains prefer a larger population size. The final decision can only be made once average plan lengths have been discussed.

Table 7.4 shows median plan length for each population size, maximum plan length pairing for each domain on a single problem. The best median lengths are highlighted in bold, unless there is not a large difference between median lengths.

- **Briefcase:** The `briefcase` domain shows large increases in median plan lengths as the maximum plan length increases. For instance, the median plan length has doubled between the maximum plan lengths of 100 to 400 candidates.
- **Blocksworld-3ops:** The `blocksworld-3ops` plans have increased in length due to the increasing maximum plan length, with only a 50% increase between 100 and 400 candidates.
- **Satellite:** The `satellite` domain presents the worst increases in plan length over all population sizes as the maximum plan length increases. The plan length doubles as the maximum plan length doubles.
- **Summary:** From the point of view of plan quality any population size can be used, but `briefcase`, `satellite`, and to a lesser extent `blocksworld-3ops` suggest either a maximum plan length of 100 or 200 actions.

Before arriving at a final number for population size and maximum plan length there are some easy decisions that can be made to make the choice of parameter values easier. The first is to rule out anything with a population size of 4000, as these make up the worst performing times for all four domains. The next step is to rule out the maximum plan lengths of 400 and 800 due to the massive increase in plan length and because the satellite problem becomes much harder when these lengths are used. This still leaves eight choices for the two values. After this the choices become slightly more subjective. At this stage it is possible to rule out a maximum plan length of 100 as 200 performs more efficiently in three of the domains except `satellite`, though there are still good times in the 200 column for that domain. If the four remaining times are ranked and summed for each domain, the setting with the lowest score is 1000/200 with 500/200 coming in a close second. 1000/200 is preferred as it performs better on the two harder domains: `blocksworld-3ops` and `satellite`. Therefore a population size of 1000 with a maximum plan length of 200 actions will be used.

| Pop | Maximum Plan Length | | | | Pop | Maximum Plan Length | | | |
|---|---|---|---|---|---|---|---|---|---|
| Size | 100 | 200 | 400 | 800 | Size | 100 | 200 | 400 | 800 |
| | Maxones | | | | | Blocksworld-3ops | | | |
| 250 | 40.07 | **1.19** | **0.97** | **0.72** | 250 | 90.77 (1) | 92.76 (2) | 115.12 (15) | 109.02 (24) |
| 500 | 30.98 | 1.92 | **1.52** | **0.98** | 500 | 98.77 (3) | 105.21 (2) | 46.41 (12) | 20.59 (10) |
| 1000 | 55.58 | 5.22 | 4.33 | 2.74 | 1000 | 84.81 (1) | **39.41** (1) | **26.38** (4) | **32.91** (2) |
| 2000 | 57.39 | 6.79 | 5.43 | 3.59 | 2000 | 97.47 | **30.98** (1) | **39.10** (2) | 56.98 |
| 4000 | 104.16 | 9.90 | 9.52 | 6.79 | 4000 | 46.03 (2) | 42.58 (1) | 55.23 (1) | 81.09 |
| | Briefcase | | | | | Satellite | | | |
| 250 | 87.24 (3) | 11.08 (1) | **4.22** | **5.90** | 250 | **66.10** | **81.49** | 150.03 (1) | 498.60 (57) |
| 500 | 86.07 (5) | **7.30** | **6.59** | **8.66** | 500 | **52.38** | **70.84** | 149.68 (2) | 499.29 (63) |
| 1000 | 39.20 (1) | 9.58 | 9.05 | 13.18 | 1000 | **79.99** | 98.27 | 277.36 (23) | 499.83 (57) |
| 2000 | 25.13 (2) | 15.77 | 16.65 | 25.39 | 2000 | 85.40 | 114.65 (1) | 211.81 (26) | 498.57 (51) |
| 4000 | 31.01 | 25.35 | 30.43 | 48.73 | 4000 | 92.07 | 103.87 (2) | 159.51 (10) | 310.28 (32) |

Table 7.3: The effect of population shape on system efficiency

| Pop Size | Maximum Plan Length 100 | 200 | 400 | 800 | Pop Size | Maximum Plan Length 100 | 200 | 400 | 800 |
|---|---|---|---|---|---|---|---|---|---|
| | Maxones | | | | | Blocksworld-3ops | | | |
| 250 | 62.00 | 62.00 | 62.00 | 62.00 | 250 | **12.00** | 15.00 | 17.00 | 21.50 |
| 500 | 62.00 | 62.00 | 62.00 | 62.00 | 500 | **12.00** | 14.00 | 17.00 | 21.00 |
| 1000 | 62.00 | 62.00 | 62.00 | 62.00 | 1000 | **12.00** | 13.00 | 16.00 | 21.00 |
| 2000 | 62.00 | 62.00 | 62.00 | 62.00 | 2000 | **12.00** | 13.00 | 16.00 | 20.00 |
| 4000 | 62.00 | 62.00 | 62.00 | 62.00 | 4000 | **11.00** | 12.00 | 15.00 | 18.00 |
| | Briefcase | | | | | Satellite | | | |
| 250 | **25.00** | 33.00 | 50.00 | 87.50 | 250 | **16.00** | 19.50 | 41.00 | 116.00 |
| 500 | **25.00** | 34.00 | 52.00 | 80.00 | 500 | **15.00** | 22.00 | 43.50 | 145.00 |
| 1000 | **25.00** | 34.00 | 51.00 | 76.50 | 1000 | **16.00** | 27.50 | 58.00 | 152.00 |
| 2000 | **26.00** | 35.00 | 51.00 | 74.00 | 2000 | **17.00** | 29.00 | 72.00 | 167.00 |
| 4000 | **26.00** | 34.00 | 51.00 | 74.00 | 4000 | **18.00** | 38.00 | 82.50 | 167.00 |

Table 7.4: The effect of population shape on plan quality

### 7.2.3 Initial Plan Length

The initial plan length specifies the amount of starting material in the population. This is done in relation to the maximum plan length, and is controlled by the initial plan length proportion. For the previous experiments this was set to 20% of the maximum plan length. For the chosen maximum plan length of 200 actions this means that each candidate is initially made up of exactly 40 actions. Though this parameter is not very important, it must be considered before Tournament Size, so that the parameters involving population initialisation can be completed.

#### 7.2.3.1 Pre-parameter Discussion

The pre-parameter experiments were performed using the follow values:

- **Initial Proportions:** 10%, 20%, 30%, 40%, 50%, 60%, 70%, 80%, 90%, 100%

The initial plan length can vary from a single action up to the entire initial candidate being filled with actions (100%). The results of the pre-parameter experiment, using `blocksworld-3ops`, showed a clear increase in efficiency by initialising the candidates with more actions for the harder problems. This gain did not continue all the way through to 100% but was replaced with degrading efficiency after 50%. More interestingly the resulting plan lengths stayed roughly constant, despite being initialised with more material. The initial plan lengths were chosen between the range of 10% and 50%. The problems for `maxones` and `briefcase` domains, are now the harder second sets and the harder sets will be used for the rest of the chapter.

- **Final Proportions:** 10% (20 actions), 20% (40 actions), 30% (60 actions), 40% (80 actions), 50% (100 actions)

#### 7.2.3.2 Results

I expect the efficiency of GENPLAN to improve as more and more material is introduced into the initial population. This could be because there are more actions in the initial population forming more diverse structures that may be required to solve the current problem. The gains will probably trail off as the initial plan length increases. Increasing the initial plan length may also have a detrimental effect on the quality of plans returned.

Figure 7.3 plots median time against initial plan length.

- **Maxones:** An initialisation of 20, 40, or 60 actions, appears acceptable for `maxones`.
- **Briefcase:** An initialisation of 60, 80, or even 100 actions appears acceptable.
- **Blocksworld-3ops:** This domain shows a good variation in efficiency. The two easiest problems have slightly degraded efficiency as initial plan lengths are increased. Whereas the two problems with the sharpest decline prefer initial plans which are longer. An initialisation of 60, 80, or even 100 actions seems acceptable.
- **Satellite:** Initial candidates with more starting material produce the best efficiency results. An initialisation of 60, 80, or even 100 actions seems acceptable.
- **Summary:** The plot shows that varying the initial plan length can bring about a slight improvement in efficiency, with `blocksworld-3ops` showing the starkest improvement in efficiency for two of the problems. The best value overall is probably 60 actions.

Figure 7.4 plots median plan length against initial plan length.

- **Briefcase:** This domain also shows fairly constant behaviour as the initial plan length increases.
- **Blocksworld-3ops:** Two of the problems show an increasing plan length as initial plan length increases while the remaining three problems remain at a constant length.
- **Satellite:** The returned plans increase in size rapidly for this domain as initial plan length increases.
- **Summary:** The results show that increasing the initial length of candidates does not have as dramatic effect on the final length of the plan as the maximum plan length parameter. The `satellite` domain indicates a preference for shorter initialisation of plans.

The choice of initial plan length is not as important as the previous two parameter choices. As long as the user does not chose very small or very large initial lengths then almost anything is acceptable. An initialisation of 60 (30%) actions appears to do well across the domains. Though for harder problems increasing the initial plan length could be advisable.

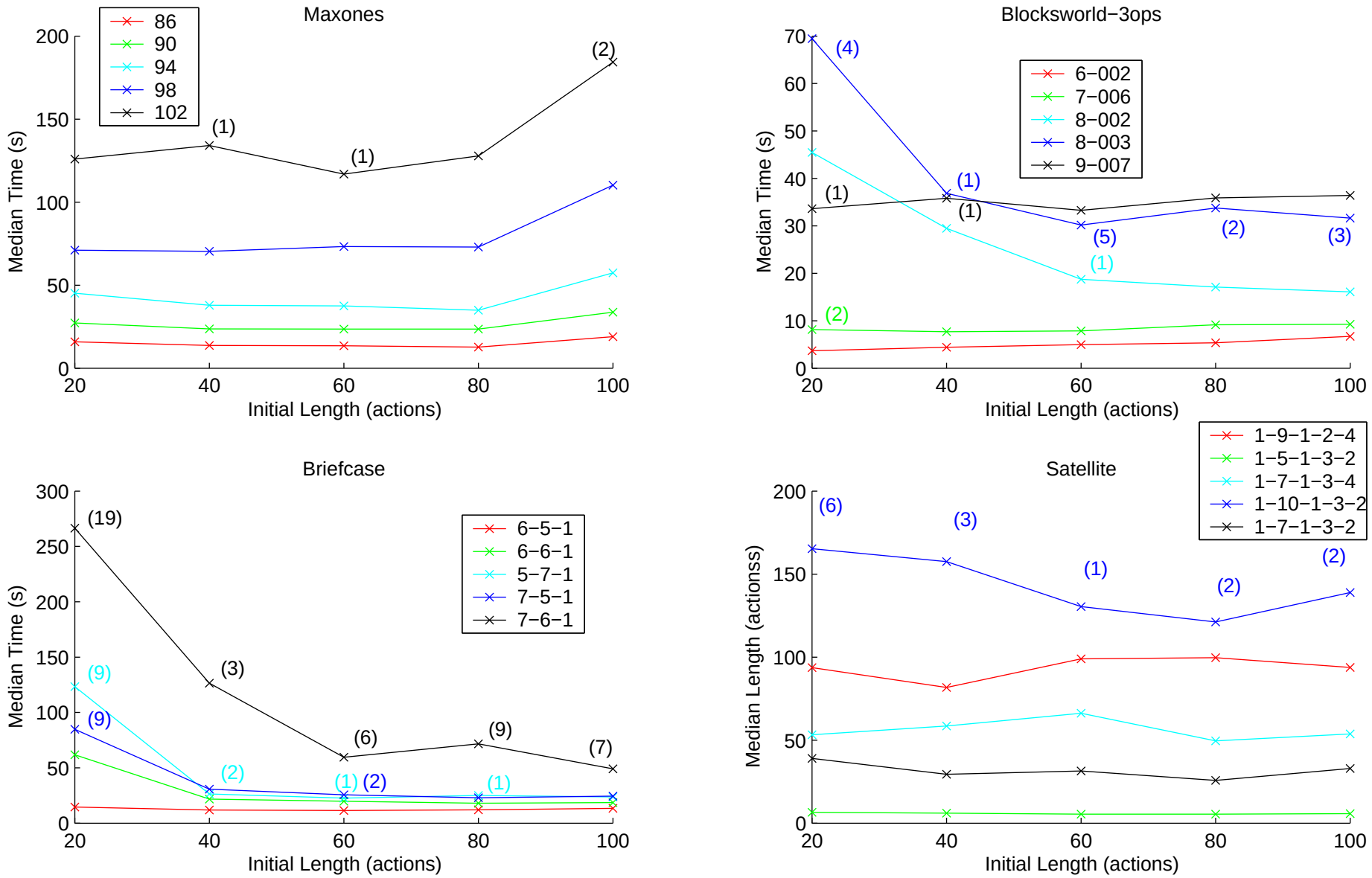



Figure 7.3: The effect of varying initial plan length on system efficiency

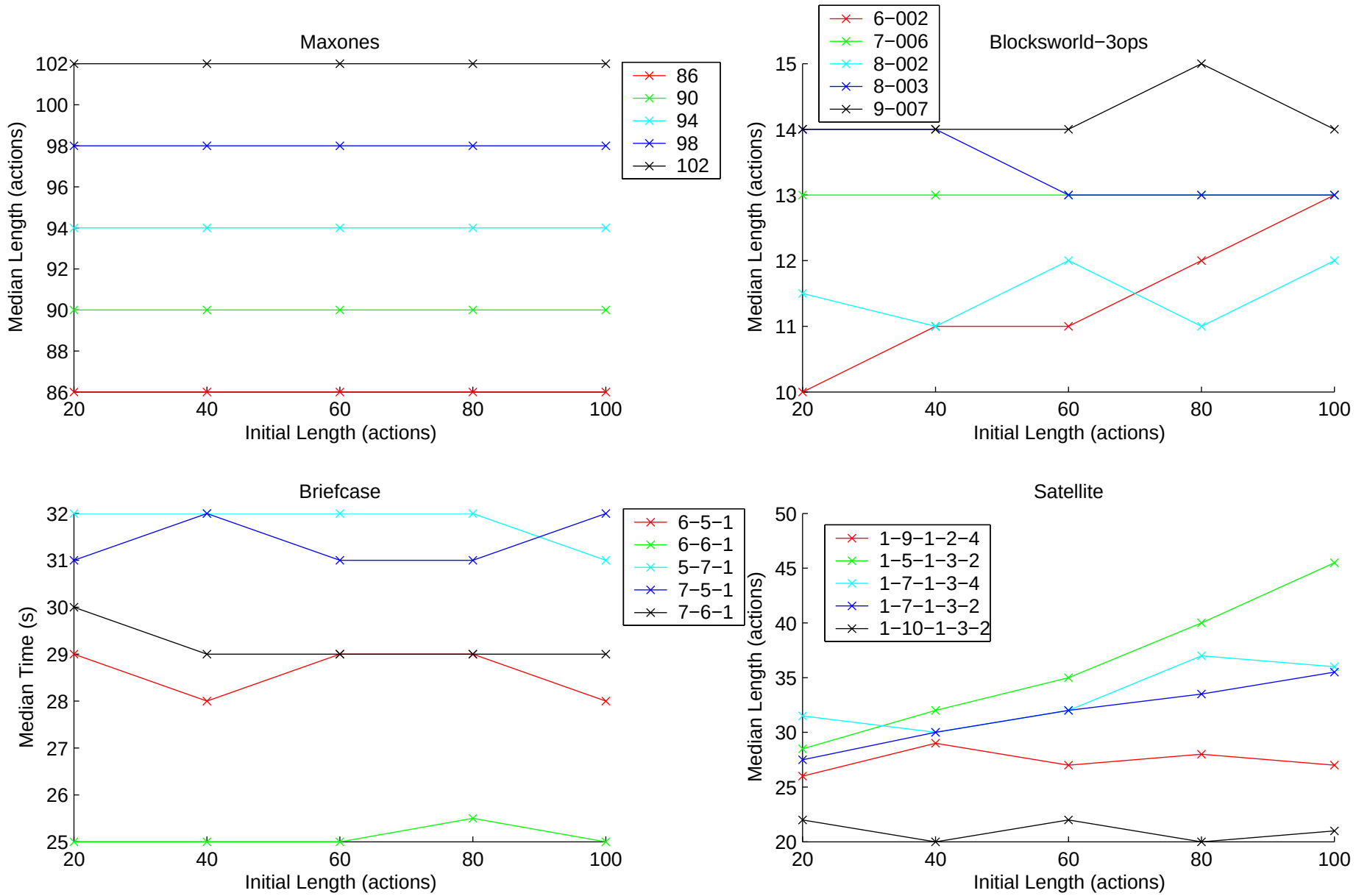



Figure 7.4: The effect of varying initial plan length on plan quality

### 7.2.4 Tournament Size

Tournament Selection is an efficient mechanism for selecting an individual for reproduction and allows for effective control of the level of selection pressure. The *Tournament Size* controls the number of individuals that undergo tournament selection. The amount of selection pressure can be varied by varying the size of the tournament. By increasing the tournament size the selection pressure goes up. After the population initialisation parameters, tournament size is next most important parameter to set as selection pressure can have a dramatic effect on efficiency and reliability.

#### 7.2.4.1 Pre-parameter Discussion

For the pre-parameter experiment I used a large range of tournament sizes:

- **Initial Sizes:** 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16

This initial size was reduced to nine values to be used in the observational experiment. The granularity for larger values was not needed as it became obvious for `blocksworld` that larger amounts of selection pressure are undesirable as GENPLAN became less reliable for the larger problems. A tournament size of 1 is included to demonstrate what happens with almost no selection pressure. There is still some slight selection pressure with the one candidate elitism. The selected tournament sizes are:

- **Final Sizes:** 1, 2, 3, 4, 5, 6, 8, 12, 16

#### 7.2.4.2 Results

The intuition for this parameter is that increasing the tournament size can make search more efficient. However, large tournament sizes increase the likelihood of failed runs, particularly for harder domains or problems. Therefore as the tournament size increases, the number of failures should increase as well, along with a decrease in efficiency. When plans are successfully found more efficiently with increased selection pressure this may result in shorter plans.

Figure 7.5 plots median solving time against tournament size.

- **Maxones:** This domain shows no detrimental effect for increasing tournament sizes. The best tournament size is 16, though the difference in improvement between 8 and 16 is small.

- **Briefcase:** The best tournament size is in the range of 2 to 4. For the hardest problem the search is less reliable after a tournament size of 8 as shown by the increasing number of failures and the increasing times. This behaviour is also present for some of the easier problems as well.

- **Blocksworld-3ops:** This domain shows many results. It supports the idea that for easy problems that a large tournament size can be used. On the other hand one problem clearly demonstrates the dangers of too much selection pressure. Also three of the problems present increasing times after a tournament size of 8. The best value is again in the range of 2 to 4.

- **Satellite:** The `satellite` results show a slight decrease in efficiency as the tournament size increases. More unexpectedly, a tournament size of 1 performs better than most. A tournament size of 1 results in almost no selection pressure.

- **Summary:** The conclusions that can be drawn from these results is that for easy problems and easy domains then the user should use a larger tournament size. But as one problem from `briefcase` and `blocksworld` demonstrate smaller tournament sizes will provide more coverage of problems at the expense of the easier problems. Smaller tournament sizes also performed better on the difficult `satellite` domain.

Figure 7.6 plots median plan length against tournament size.

- **Briefcase:** The quality of returned plans is unaffected by the different tournament sizes. The only conclusion here is that any tournament size is acceptable. What is also interesting is the average plan length is not increasing as the median time increases.

- **Blocksworld-3ops:** The quality of returned plans is unaffected by the different tournament sizes.

- **Satellite:** This domain has been resistant to any changes in parameters but there is a clear downwards trend for median plan length resulting from increased selection pressure. This is despite the fact that the plans are on average taking longer to find as selection pressure increases.

- **Summary:** The tournament size appears to have little affect on the average length of the returned plan, but the exception to this is `satellite` showing a small downwards trend using larger tournament sizes.

A final decision will not be taken on tournament size until the behaviour of GENPLAN is analysed when varying the probability of crossover concurrently with tournament size. The best tournament sizes at this stage are 2, 3, 4, and this is with a probability of crossover at 80%. An investigation into the two parameters together may be interesting as Goldberg suggests a strong relationship between tournament selection and probability of crossover [Goldberg, 2002].

### 7.2.5 Probability of Crossover and Tournament Size

This observational experiment examines the relationship between probability of crossover and tournament size. When creating the next population the first step is select whether the children to be inserted into the next population come form reproduction or crossover. This is controlled by the ratio between crossover and reproduction. The default for this is 80% probability of crossover and 20% probability of reproduction, or in another way 80/20.

#### 7.2.5.1 Pre-parameter Discussion

The pre-parameter experiments used the following ratios:

- **Initial Ratios:** 20/80, 30/70, 40/60, 50/50, 60/40, 70/30, 80/20, 90/10, 100/0

At the left end of the spectrum of values GENPLAN is relying almost on minimal amounts of crossover to solve the problem. This may suggest that mutation has a greater role to play in such a scenario. An interesting side experiment would be to examine whether a mutation-based framework is better than a crossover-based framework. The efficiency of GENPLAN decreases as the amount of crossover is decreased for the harder `blocksworld` problems. The tournament size used for the pre-parametrisation experiments was set to 2. The tournament size values 5 and 6 are used to incorporate a larger study between tournament size and probability of crossover. The selected values for this experiment are:

- **Final Ratios:** 20/80, 40/60, 60/40, 80/20, 100/00
- **Final Tournament Sizes:** 2, 3, 4, 5, 6

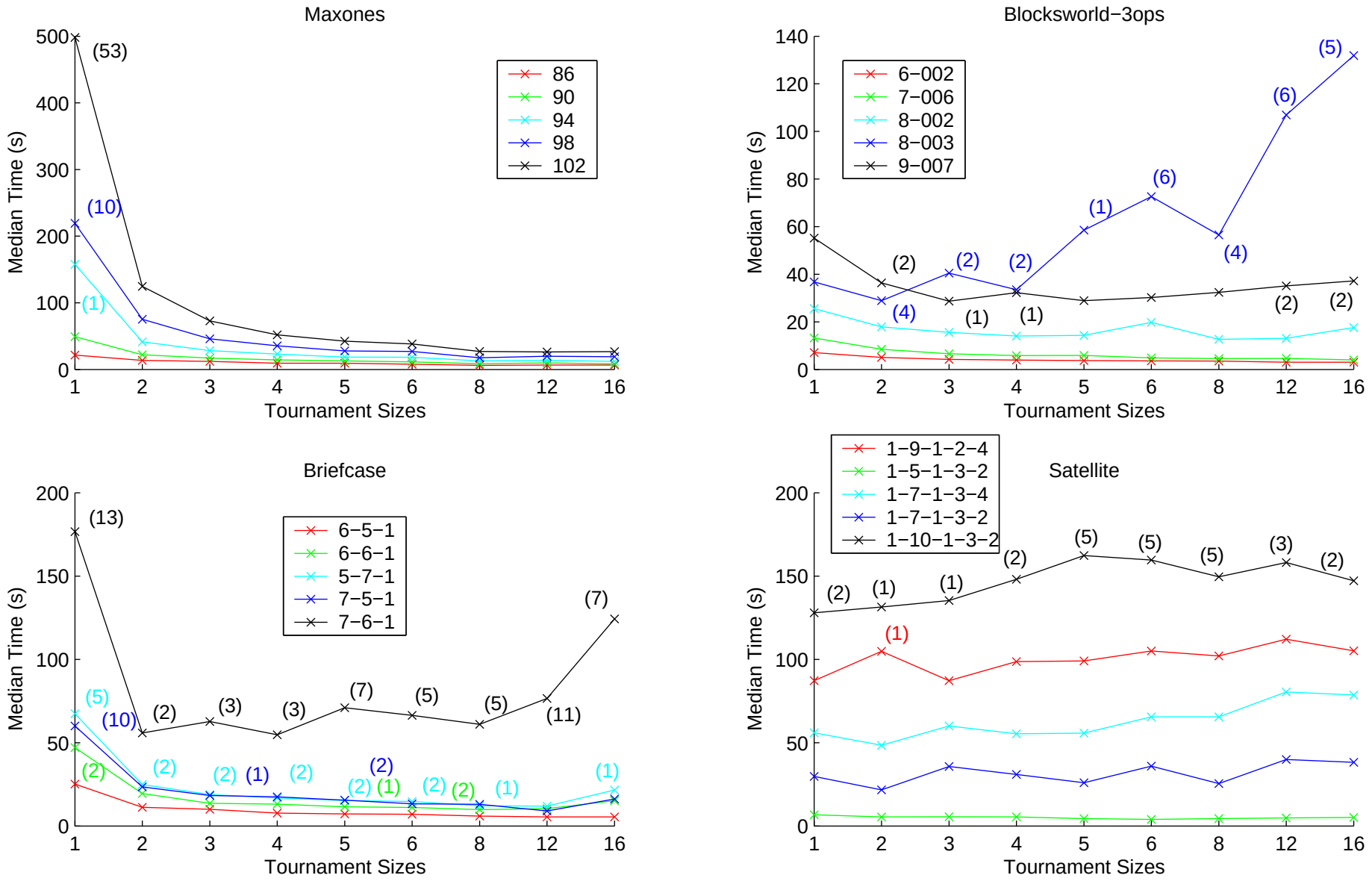



Figure 7.5: The effect of varying tournament size on system efficiency

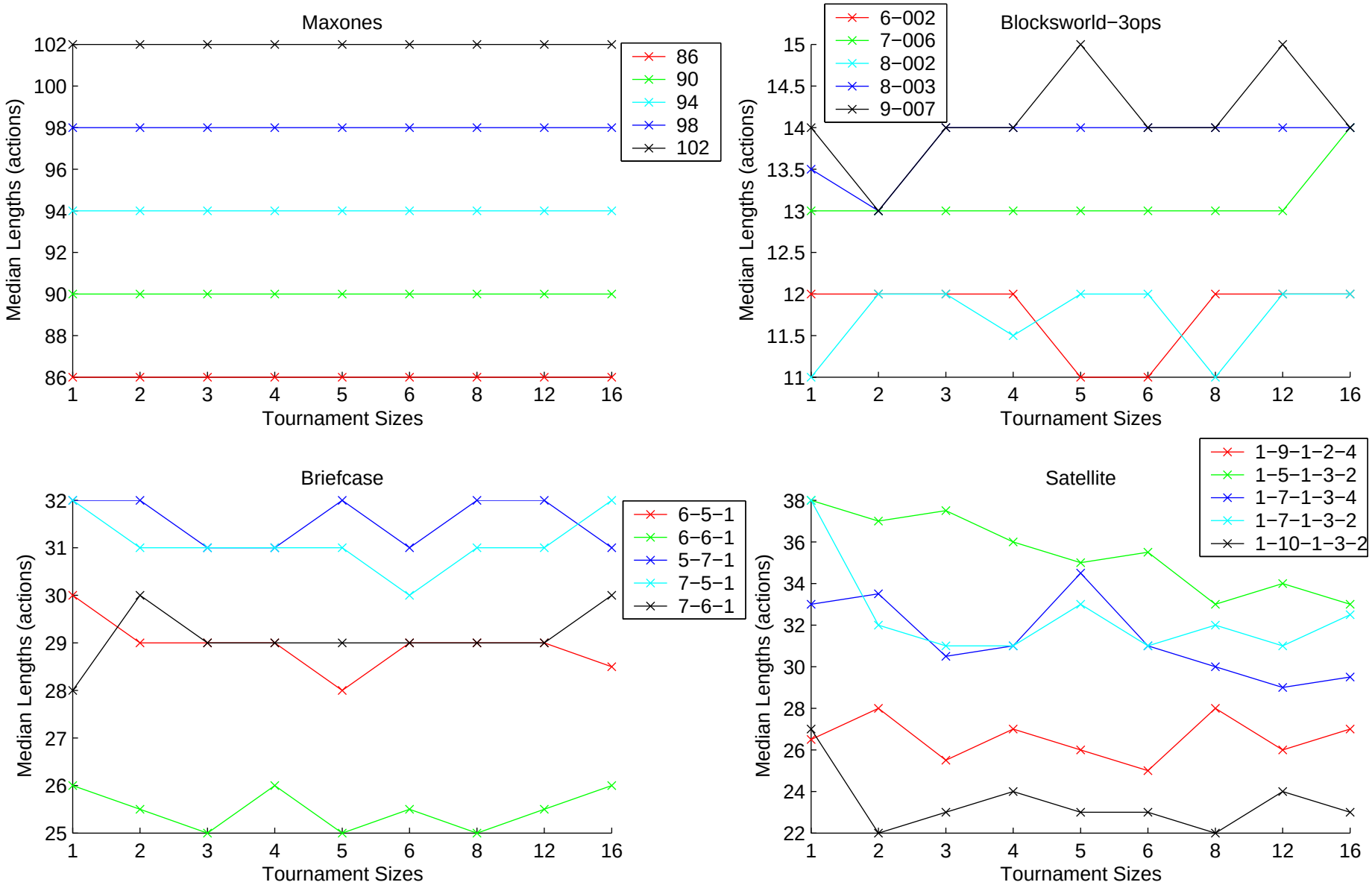



Figure 7.6: The effect of varying tournament size on plan quality

#### 7.2.5.2 Results

The intuition for this experiment is that small sizes of tournament selection combined with small amounts of crossover should be more inefficient than when using large tournament sizes and large amounts of crossover. Though the relationship will not be strictly linear. I would expect the plan quality to vary in the same way, with lowest quality plans being produced when a low probability of crossover tournament size are used and and the best quality plans being produced at the opposite end. If there is any discernible variation at all.

Table 7.5 reports the median time against tournament size ($s$) and probability of crossover ($P_c$).

- **Maxones:** The best values are around the setting 6/20, that is large tournament sizes with small amounts of crossover.
- **Briefcase:** The best values are around the setting 6/100, that is large tournament sizes with large amounts of crossover.
- **Blocksworld-3ops:** The best values are around the setting 3/80, that is a medium sized tournament, with a mixture of crossover and reproduction.
- **Satellite:** The best values are around the setting 3/100, that is a medium sized tournament and a large amount of crossover.
- **Summary:** For three out of four domains larger tournament sizes are to be preferred with large amounts of crossover. Only `maxones` is the exception to this rule, as it prefers a large tournament size with a small amount of crossover.

Table 7.6 reports the median length against tournament size ($s$) and probability of crossover ($P_c$).

- **Briefcase:** Plan quality seems unaffected by these parameters.
- **Blocksworld-3ops:** Again for this domain plan quality appears unaffected by the various settings.
- **Satellite:** This domain shows a bit more variation in plan quality but with no discernible pattern.
- **Summary:** The quality of plans seems unaffected by these parameters, thus leaving efficiency as the primary concern.

Taking these results on their own it would appear that a large amount of selection pressure with a high percentage of crossover is the preferred option. But as the experiments in the previous subsection show, values of tournament size above 4 are undesirable. This leaves the choice for these two parameters to be quite subjective. In order to increase the efficiency of GENPLAN I opted to increase the tournament size from 2 to 3 as it does not appear to have too much impact on reliability. As three of the harder domains all appear to prefer large amounts of crossover a value of 80% or 100% seems sensible. I believe some reproduction is necessary in an evolutionary system, in order to maintain some discovered partial solutions. I choose to persist with the 80% value. Thus making a tournament size of 3 and 80% crossover the final values.

| $s$ | $P_c$ | | | | | $s$ | $P_c$ | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | 20 | 40 | 60 | 80 | 100 | | 20 | 40 | 60 | 80 | 100 |
| | Maxones | | | | | | Blocksworld-3ops | | | | |
| 2 | 32.02 | 36.10 | 52.20 | 71.14 | 84.95 | 2 | 156.83 (8) | 99.03 (2) | **29.15** (5) | **37.55** (4) | **37.32** (4) |
| 3 | 27.70 | 30.27 | 36.25 | 43.25 | 60.45 | 3 | 148.89 (6) | 154.43 (10) | 130.59 (6) | **23.64** (4) | **28.46** (2) |
| 4 | **24.16** | 24.80 | 30.22 | 32.95 | 41.83 | 4 | 199.94 (9) | 106.32 (5) | 100.26 (4) | 67.98 (5) | 40.96 (1) |
| 5 | **22.74** | 26.25 | 25.36 | 26.37 | 34.77 | 5 | 176.09 (3) | 145.88 (8) | 149.72 (7) | 70.27 (7) | 79.52 (3) |
| 6 | **21.67** | **21.91** | **23.96** | 26.13 | 27.57 | 6 | 192.25 (6) | 151.13 (2) | 59.53 (4) | 50.92 (5) | 93.49 (3) |
| | Briefcase | | | | | | Satellite | | | | |
| 2 | 109.86 (10) | 15.87 | 25.34 (1) | 23.92 (1) | 23.13 (1) | 2 | 222.48 (9) | 141.94 (1) | 103.83 | **97.63** | **96.11** |
| 3 | 79.97 (12) | 34.23 (2) | 25.64 (4) | 19.73 | 18.47 (2) | 3 | 188.06 (4) | 143.27 (2) | 115.43 | **90.40** | 97.78 |
| 4 | 77.48 (5) | 33.38 (6) | **15.42** (2) | **15.43** | 19.37 (1) | 4 | 199.97 (7) | 139.03 (1) | 110.93 | **92.32** | **75.52** |
| 5 | 103.98 (3) | 25.64 (1) | 19.19 (2) | 15.47 | 15.50 (1) | 5 | 213.08 (3) | 171.42 | 124.07 | 113.28 (1) | 139.72 (1) |
| 6 | 142.82 (10) | 47.04 (5) | **14.72** (1) | **13.52** | **10.12** | 6 | 218.98 (5) | 165.98 | 112.47 | 116.78 | 109.16 |

Table 7.5: The effect of varying tournament size and probability of crossover on system efficiency

| $s$ | $P_c$ | | | | | $s$ | $P_c$ | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | 20 | 40 | 60 | 80 | 100 | | 20 | 40 | 60 | 80 | 100 |
| | Maxones | | | | | | Blocksworld-3ops | | | | |
| 2 | 98 | 98 | 98 | 98 | 98 | 2 | 13.00 | 14.00 | 14.00 | 14.00 | 13.00 |
| 3 | 98 | 98 | 98 | 98 | 98 | 3 | 14.00 | 13.50 | 14.00 | 14.00 | 14.00 |
| 4 | 98 | 98 | 98 | 98 | 98 | 4 | 14.00 | 14.00 | 15.00 | 14.00 | 14.00 |
| 5 | 98 | 98 | 98 | 98 | 98 | 5 | 14.00 | 15.00 | 15.00 | 14.00 | 14.00 |
| 6 | 98 | 98 | 98 | 98 | 98 | 6 | 14.00 | 14.00 | 14.00 | 13.00 | 14.00 |
| | Briefcase | | | | | | Satellite | | | | |
| 2 | 31.00 | 30.00 | 31.00 | 31.00 | 32.00 | 2 | 29.00 | 25.00 | 27.00 | 25.00 | 26.00 |
| 3 | 31.00 | 30.00 | 31.00 | 32.00 | 31.00 | 3 | 30.00 | 26.50 | 27.00 | 27.50 | 29.00 |
| 4 | 31.00 | 31.00 | 31.00 | 31.00 | 32.00 | 4 | 26.00 | 25.00 | 25.50 | 30.00 | 26.00 |
| 5 | 30.00 | 31.00 | 31.00 | 31.00 | 32.00 | 5 | 29.00 | 25.00 | 26.50 | 27.00 | 23.00 |
| 6 | 31.00 | 31.00 | 32.00 | 31.50 | 31.00 | 6 | 31.00 | 26.00 | 25.00 | 24.00 | 27.00 |

Table 7.6: The effect of varying tournament size and probability of crossover on plan quality.

### 7.2.6 Variable Add Mutation

Though Koza originally suggested not to use mutation with GP-based systems [Koza, 1992], differing mutation operators are widely used in the GA and GP communities. Mutation can have a beneficial affect on EAs in general, by bringing extinct elements back into a population that may have been lost during a run. To this end, the `VarAddMutation` operator simply adds a random number of random actions to random positions of a child (either from reproduction or crossover). The type and frequency of mutation can also have an effect on performance, but this particular mutation is fairly weak. The `VarAddMutation` operator is examined by varying parameters that increase both amount and frequency.

#### 7.2.6.1 Pre-parameter Discussion

The pre-parameterisation experiment was skipped and the chosen values for the frequency and range of additions are listed below. This could be done as the final set of values chosen are quite broad.

- **Frequency** 10%, 20%, 30%, 40%, 50%, 60%, 70%
- **Number of actions** 0-10, 0-20, 0-30, 0-40, 0-50, 0-60

#### 7.2.6.2 Results

The intuition is that increasing the amount of mutation will help GENPLAN solve problems more efficiently, however the effects will not be as noticeable as other parameters. As candidates within the population increase in length the actual number additions that can be made will fall well below the maximum specified by the mutation amount. This may partly explain why the operator may not have much affect on efficiency and plan quality. As this mutation operator actively increases the lengths of plans I expect the average plan length to increase as the frequency and number of added actions increase.

Table 7.7 reports median time against different frequencies and amounts of mutation.

- **Maxones:** The results indicate for this domain it is best to use low levels of mutation, in particular low frequency. The best values are in the 10% row.
- **Briefcase:** This domain also prefers a low level of frequency, with the best values in the top left hand corner, around the 10% 0-10 actions setting.

- **Blocksworld-3ops:** This domain suggests that low frequencies of mutation and low amounts should be used. The best values are around 10% frequency, 0-20 actions.

- **Satellite:** The results for this domain are very spread out. In comparison to the other three domains larger amounts of mutation are to be preferred and the best values are around the centre of the table.

- **Summary:** The mutation operator is one of the few operators to have a large effect on the efficiency of GENPLAN on `satellite`. For the other domains low-levels of mutation appear beneficial.

Table 7.8 reports median plan length for different frequencies and amounts of mutation.

- **Briefcase:** The mutation operator had no affect on the median length of the plans for this domain.

- **Blocksworld-3ops:** Again the operator had no affect on the final median length of the returned plan for this domain.

- **Satellite:** The only domain to show a definite trend for preferring less mutation as the quality of the returned plans degrades as a result of increased mutation.

- **Summary:** Unexpectedly, only satellite produced longer plans on average for increasing amounts of mutation.

Choosing the best mutation settings from these results is difficult as the mutation operator did not have as much impact on performance. Table 7.9 shows that leaving the mutation switched off is detrimental. For the more difficult domains, `satellite`, and `blocksworld` the mutation operator has clear benefits. As there is no clear winner I decided on 20% and 0-30 actions, which is more aggressive than the original setting.

| *Freq* | *Amount* | | | | | | *Freq* | *Amount* | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 0-10 | 0-20 | 0-30 | 0-40 | 0-50 | 0-60 | | 0-10 | 0-20 | 0-30 | 0-40 | 0-50 | 0-60 |
| | Maxones | | | | | | | Blocksworld-3ops | | | | | |
| 10% | 46.50 | **46.15** | 49.65 | **44.96** | **46.46** | **46.45** | 10% | 42.04 | 25.29 (1) | 27.66 (1) | 26.50 | 24.04 | **23.11** |
| 20% | 48.79 | **46.32** | 50.33 | 51.28 | 50.81 | 51.89 | 20% | 36.69 | **19.38** | 24.08 | 24.12 | 25.58 | 26.63 |
| 30% | 48.54 | 51.78 | 56.24 | 55.35 | 60.21 | 57.98 | 30% | 24.40 | 26.28 | 23.34 | 23.25 | 23.41 | 27.82 |
| 40% | 62.64 | 58.09 | 53.34 | 56.26 | 60.93 | 61.58 | 40% | 32.81 | **22.99** | 24.92 | 39.02 | 41.90 | 33.33 |
| 50% | 58.86 | 56.41 | 62.96 | 66.07 | 57.17 | 63.73 | 50% | 25.98 | **22.02** | 28.27 | 27.46 | 32.51 | 30.05 |
| 60% | 56.65 | 62.11 | 63.12 | 49.79 | 47.20 | 55.09 | 60% | 24.55 | 28.04 | 27.14 | 30.96 | 34.29 | 35.28 |
| 70% | 59.39 | 64.86 | 63.95 | 61.26 | 66.89 | 72.62 | 70% | 31.77 | 33.04 | 35.53 | 34.70 | 33.57 | 39.14 |
| | Briefcase | | | | | | | Satellite | | | | | |
| 10% | **19.87** (1) | **19.12** | 20.08 | **20.05** | 20.37 | 20.61 | 10% | 101.59 | 87.06 | 75.06 | 77.61 | 75.56 | 75.07 |
| 20% | **18.29** | 21.07 | **18.93** | 22.32 | 20.80 | 21.22 | 20% | 69.47 | 81.80 | 70.52 | 76.08 | 66.77 | 69.71 |
| 30% | 20.29 | 22.22 | 21.85 | 22.75 | 23.39 | 25.61 | 30% | 78.33 | **63.05** | 66.35 | 70.79 | 66.71 | 74.32 |
| 40% | 22.34 | 22.57 | 23.45 | 24.32 | 27.81 | 26.87 | 40% | 75.29 | 66.60 | 69.64 | 69.22 | 65.45 | **64.73** |
| 50% | 23.66 | 23.88 | 27.20 | 30.11 | 29.93 | 32.97 | 50% | 77.79 | 67.95 | **59.08** | 71.73 | **62.99** | 73.67 |
| 60% | 23.45 | 28.38 | 29.82 | 30.28 | 36.17 | 32.49 | 60% | 79.69 | 74.13 | 79.60 | 65.85 | 89.02 | 76.23 |
| 70% | 25.67 | 29.41 | 35.15 | 31.00 | 33.15 | 37.60 | 70% | 70.30 | 73.11 | **65.01** | 78.01 | 76.00 | 76.84 |

Table 7.7: Efficiency results with variable amounts of mutation

| *Freq* | *Amount* | | | | | | *Freq* | *Amount* | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 0-10 | 0-20 | 0-30 | 0-40 | 0-50 | 0-60 | | 0-10 | 0-20 | 0-30 | 0-40 | 0-50 | 0-60 |
| | Maxones | | | | | | | Blocksworld-3ops | | | | | |
| 10% | 98 | 98 | 98 | 98 | 98 | 98 | 10% | 14.00 | 13.00 | 14.00 | 14.00 | 13.00 | 13.00 |
| 20% | 98 | 98 | 98 | 98 | 98 | 98 | 20% | 13.00 | 13.00 | 13.00 | 13.00 | 13.00 | 13.00 |
| 30% | 98 | 98 | 98 | 98 | 98 | 98 | 30% | 14.00 | 13.00 | 13.00 | 13.00 | 13.00 | 13.00 |
| 40% | 98 | 98 | 98 | 98 | 98 | 98 | 40% | 13.00 | 13.00 | 13.00 | 12.00 | 13.00 | 12.00 |
| 50% | 98 | 98 | 98 | 98 | 98 | 98 | 50% | 13.00 | 12.00 | 12.00 | 13.00 | 12.00 | 13.00 |
| 60% | 98 | 98 | 98 | 98 | 98 | 98 | 60% | 12.00 | 13.00 | 13.00 | 12.00 | 12.00 | 12.00 |
| 70% | 98 | 98 | 98 | 98 | 98 | 98 | 70% | 13.00 | 13.00 | 12.00 | 12.00 | 12.00 | 12.00 |
| | Briefcase | | | | | | | Satellite | | | | | |
| 10% | 31.00 | 32.00 | 32.00 | 31.00 | 31.50 | 32.00 | 10% | **27.00** | **29.50** | **32.00** | 33.00 | 33.50 | 33.00 |
| 20% | 31.50 | 32.00 | 31.00 | 31.00 | 31.00 | 30.00 | 20% | **30.00** | **32.00** | 34.00 | 33.00 | 36.00 | 34.00 |
| 30% | 31.00 | 32.00 | 31.00 | 31.00 | 31.00 | 32.00 | 30% | **32.00** | 34.50 | 38.00 | 34.50 | 38.50 | 37.00 |
| 40% | 31.00 | 32.00 | 31.00 | 32.00 | 31.00 | 31.00 | 40% | 35.00 | 33.00 | 36.50 | 38.00 | 39.00 | 39.00 |
| 50% | 31.00 | 31.00 | 31.00 | 31.00 | 30.00 | 31.00 | 50% | 35.50 | 36.00 | 39.00 | 41.00 | 39.00 | 40.00 |
| 60% | 31.00 | 31.00 | 30.00 | 30.00 | 31.00 | 30.00 | 60% | 35.50 | 37.50 | 39.50 | 37.00 | 42.00 | 42.00 |
| 70% | 31.00 | 30.00 | 30.00 | 30.50 | 31.00 | 30.00 | 70% | 37.00 | 37.00 | 39.00 | 38.00 | 39.50 | 39.00 |

Table 7.8: Plan quality results with variable amounts of mutation

| Domain | Median Time (Failures) | Median Length |
|---|---|---|
| Maxones | 39.53 (1) | 98 |
| Briefcase | 18.61 (19) | 33 |
| Blocksworld | 45.72 (46) | 14 |
| Satellite | 497.30(80) | 30 |

Table 7.9: Mutation switched off

## 7.3 GENPLAN Parameters

This chapter concludes with a table showing the original and final chosen values for each of the parameters, Table 7.10. On the whole, the new parameters represent a slightly more aggressive approach with a larger maximum plan length, a larger tournament size, and more mutation.

## 7.4 Chapter Summary

The main purpose of this chapter was to devise a plausible parameterisation of GENPLAN that will be used to create the core results of GENPLAN and to form a control setting with which to compare against with any implemented improvements to the algorithm. The parameterisation was achieved by running a series of observational experiments on the fundamental parameters over a necessarily small subset of domains and problems. The behaviour of EAs can be improved through careful manipulation of parameters but this is not the final answer in designing a scalable algorithm which has adopted a basic approach. The behaviour of GENPLAN is also fairly robust in relation to assigned parameters, if extreme values are ignored. The next chapter uses the selected parameterisation to configure the algorithm and to report results from applying GENPLAN to a much larger corpus of domains and problems.

| Parameter | Original Setting | Final Setting |
|---|---|---|
| Population Size | 1000 | 1000 |
| Maximum Plan Length | 100 | 200 |
| Initial Actions | 20% | 30% |
| Termination Criteria | Max. Time and Max. Fitness | Max. Time and Max. Fitness |
| Tournament Size | 2 | 3 |
| Probability of Crossover | 80% | 80% |
| Var Add Mutation | 10%, 0-8 actions | 20%, 0-30 actions |

Table 7.10: Original and Final Settings

# Chapter 8

# GENPLAN Results and Evaluation

This chapter presents the results of applying GENPLAN to a selection of classical planning domains. The behaviour of GENPLAN is recorded using a series of observational experiments. For each domain, I generated a new set of problems, either by using existing random problem generators or using my own random problem generators. Some domains include results from pre-existing problem sets already available to the planning community, in particular problems created for past international planning competitions. The performance of GENPLAN is measured against the problems using recognised metrics. The behaviour is subsequently commented upon in relation to the known search issues of the domain in question. The performance of GENPLAN is also evaluated by running a set of planners on the same problem sets. The evaluation of GENPLAN is performed using the non-linearity test to examine different scaling behaviours. The next chapter examines the behaviour of GENPLAN in relation to specific search issues and puts forward reasons for explaining GENPLAN's performance.

This chapter forms the core of the thesis. Once the basic algorithm has been designed, implemented, and configured it needs to be evaluated against a corpus of domains. The goal of this is to both investigate GENPLAN's behaviour on a selection of domains and to also evaluate the behaviour in terms of other planners. Chapter 4 showed that EA-based planning systems have only been previously applied to small selections of planning domains, and to domains that are not well recognised by the planning community. This motivates the need to apply the approach to a range of domains and problems, especially as most previous approaches have reported positive results with little comparison to standard planning technology. The results of this chapter will highlight any strengths and weaknesses of the technique in relation to easy and difficult planning domains. This chapter investigates the research question of how well

this EA-based approach performs at classical planning.

## 8.1 Experimental Methodology

The experimental methodology proposed in Chapter 6 is adopted and modified for this chapter. The experiments are divided into two parts. The first part is an initial examination of all the planners performance on the test domains. The second part more closely examines the performance of GENPLAN as compared to the test planners with the aid of the non-linearity test.

### 8.1.1 Choose Goals

The purpose of these experiments is to investigate the behaviour of GENPLAN over a range of domains in order to present its performance through a series of observational experiments. The performance of GENPLAN can be measured by how quickly it solves problems over a range of selected domains and the quality of plans produced. A second goal of these experiments is to evaluate the performance of GENPLAN against a selection of test planners by running the planners on the same problems. The evaluation will be done in terms of scaling behaviour rather than raw efficiency results. This is because GENPLAN is not a competition planner and has not been heavily optimised and its raw efficiency results will be poor in comparison with the other planners. However, it should be stated that the scaling evaluation was a post-hoc analysis and was performed after the problems were chosen and the results were collected. The domains have been selected in relation to: dead-ends, number of operators, maximum size and number of local minima, minimum bench size, and action space. The performance of GENPLAN is related back to these known search issues.

### 8.1.2 Choose Metrics

The median CPU time will be used to measure efficiency and median plan length is used to measure plan quality, as specified in Chapter 6.

### 8.1.3 Choose Problems

This section is again broken down into domain choice and problem choice.

| Domain | Dead-ends | Number of Actions | Max Local Minima | Min Bench Size |
|---|---|---|---|---|
| Maxones | harmless | 1 | 0 | 1 |
| TTD | harmless | 2 | 1 | 1 |
| Briefcase | undirected | 3 | 1 | 3 |
| **Gripper** | undirected | 3 | 1* | 3 |
| Blocksworld-3ops | undirected | 3 | $< c$* | $< c$ |
| Numeric Mystery | unrecognised | 3 | 1 | 3 |
| **Miconic-STRIPS** | harmless | 4 | 1 | 3 |
| **Satellite** | harmless | 5 | 1* | 3 |
| **DriverLog** | undirected | 6 | 1* | 4 |

Table 8.1: The selected domains

#### 8.1.3.1 Choosing Domains

The domains are chosen to form a representative sample over the available planning domains. This can be achieved by using the dimensions, in relation to search and other metrics, established in Chapters 2 and 6. In addition to these metrics the action space is given for each domain as it appears in the results section. The action space metric is the number of instantiated actions given a particular domain-problem pairing and can be calculated mathematically. The size of the action space helps define the size of the search space. All domains presented in Chapter 6 are used for this chapter and their characterisation in terms of search is repeated in Table 8.1. The domains are given in full in Appendix A. The domains listed in the table represent a good sample of typed STRIPS domains and are presented in order of increasing difficulty.

#### 8.1.3.2 Generating Problems

The way in which problems have been generated has been rigorous as well. Instances from a domain can form particular subclasses of the original domain, with different properties to a typical problem of a domain. The problems created for the `blocksworld` domain for IPC 2 are a good example of this, as they all come from one class of problems involving multi-stack to single stack problems. There are also dangers in using pre-established benchmark sets as well. One benchmark set created for `blocksworld`, called large-* (where * stands for a letter), has been criticised for being non-typical [Slaney and Thiébaux, 2001]. In order to overcome this new instances are

created randomly with random problem generators and problems are explicitly defined. On the other hand some existing problem sets from previous planning competitions are also used.

Care must be taken in generating random instances for domains. Firstly problems for domains can vary in size, for example changing the number of blocks in `blocksworld`. Each domain is specified in terms of the number of objects in the problem, which numbers of objects are varied, and how they were varied. Specific ranges of difficulty of problems to domains were found through assessment experiments using larger sets of problems. Further, values for search issues can be redefined depending on how problems are created. For instance, the domain engineer can rule out local minima in the `briefcase` domain if a goal location for the briefcase is not specified. All these decisions are specific to a domain and they are summarised when the domain is discussed. All the selected problems are assumed to be valid instances and solvable. Invalid and unsolvable problems will only create ceiling effects as there is no explicit mechanism in GENPLAN to deal with unsolvable instances. An unsolvable instance will cause GENPLAN to fail to plan and produce an uninteresting result. Links to my own random problem generators, those generators written by others, and problem sets are available online[1].

### 8.1.4 Designing the Observational Experiments

Once the domains and problems have been selected, it is necessary to run the experiment. GENPLAN was run 100 times for each problem using the parameter settings decided upon in Chapter 7. The machine platform, the actual mechanics of running the experiments, and the way in which metrics were recorded was presented in Chapter 6. For completeness the parameters are repeated in Table 8.2.

### 8.1.5 Designing the Manipulation Experiments

In order to carry out the evaluation of GENPLAN a group of test planners need to be selected so that GENPLAN can be compared to other planners. Four planners were selected in order to help evaluate the performance of GENPLAN. Each planner was chosen to represent a type of search: partial order planning, graphplan planning, SAT-planning, and heuristic search. Descriptions of these types of search were presented in Chapter 2. Ideally "vanilla" versions of each type of search would be used to compare

[1] http://homepages.inf.ed.ac.uk/s9638800/index.html

| Parameter | Setting |
|---|---|
| Number of Runs | 100 |
| Maximum Time | 500 seconds |
| Population Size | 1000 |
| Maximum Plan Length | 200 |
| Initial Actions | 30%, or 60 actions |
| Termination Criteria | Max. Time and Max. Fitness |
| Tournament Size | 3 |
| Probability of Crossover | 80% |
| Var Add Mutation | 20%, 0-30 actions |

Table 8.2: Standard Parameters

against the basic approach of GENPLAN. Such planners do not exist so a compromise was performed and a set of planners were chosen to be good planners for their style of search and which have all appeared in previous planning competitions.

- VHPOP [Younes and Simmons, 2003]: This is a partial order partial planner that won the "Best Newcomer" prize at IPC 3. It is a very flexible planner that implements multiple heuristics and flaw selection strategies. The planner was run using the same parameterisation as used in IPC 3. VHPOP is a sub-optimal planner.

- IPP4 [Koehler et al., 1997]: This is a graphplan-based planning system that was submitted to both IPC 1 and IPC 2. IPP4 was run using the default parameterisation. The planner takes very few parameters that could affect performance. As IPP4 is a graphplan-based planner and it produces optimal parallel plans. This planner is referred to as IPP in the discussion.

- Blackbox [Kautz and Selman, 1999]: This a planning system based on solving SAT instances produced by planning problems using stochastic local search. The planner can also use mixed strategies by incorporating a graphplan solver and various local search algorithms, but for the results in this chapter Blackbox was restricted to only using the "chaff" solver. This solver is presented as the best stochastic solver. Blackbox appeared in IPC 1 and is sub-optimal when only a

stochastic solver is used as the stochastic solvers do not guarantee completeness. This planner is referred to as BB in the discussion.

- Metric-FF [Hoffmann, 2003]: This is a heuristic planner that uses forward hill-climbing search in conjunction with the $h^+$ heuristic. Metric-FF has won two awards at previous planning competitions. Metric-FF uses part of the graphplan planning algorithm but only as an aid to search and not as a complete problem solving strategy. The planner produces sub-optimal plans. This planner is referred to as FF in the discussion.

The results presented here should not be interpreted as a competition but more as an examination of behaviour of planners over a range of domains. The planners were each run on machines identical to those specified in Chapter 6. Similarly to GENPLAN, all planners were given 500 seconds processing time and 512 megabytes of memory with which to find a solution to a problem. Occasionally, in order to generate more results, these constraints were relaxed to 3000 seconds and 2Gb of memory. All planners were run 100 times for each problem. This was done to reduce the amount of variability in the timing results.

The goal of the evaluation is to determine whether GENPLAN's scaling behaviour is better or worse than any of the other test planners using the data collected for the observational experiments. The scaling behaviour of the four test planners in relation to GENPLAN's own scaling behaviour is evaluated with the aid of the non-linearity test presented in Chapter 6. The examination was performed after the data was collected and hypotheses are reverse engineered from the plots describing the behaviour of all five planners on the random problem instances.

The non-linearity test needs to be applied with care to the data produced by the five planners on the domains. There were three main issues that were taken into account before hypotheses were created in order to test the data: ceiling effects, grouping effects, and dominance effects.

Ceiling effects are apparent when GENPLAN has failed a large number of times (around 85 failures) on a particular instance. This results in a median time that is an underestimate of the true median time as the program was not allowed to run to completion due to the 500 second time limit. This then distorts the visual scaling behaviour of GENPLAN by flattening the resulting curve unrealistically.

Grouping effects occur due to the way instances are selected for some domains. For instance, when selecting problems for the `TTD` domain the number of cities has

been fixed at various intervals. This can sometimes produce multiple distinct lines in the scatterplots when all the data is used. It would be better to analyse the results from each specific sized set individually.

Dominance effects are related to grouping effects. If results are collected from several different problem sets with quite different levels of difficulty the results in the hardest problem set can dominate the other results. In that, results from easier problems are clustered almost into a single point in the resulting scatterplot. This last problem is a result of how problems are sampled for some of the domains.

Where any of the above are an issue for a particular domain it will be reported in the main body of text and hypotheses will be selected appropriately. Generally this means hypotheses are generated for particular sets of data or when it is sensible to do. Otherwise, the results from different sets are collected into a single set and tested.

### 8.1.6 Presenting the Results

The standard graphs and tables described in Chapter 6 are used to report results from the experiments. Full tables of all results are available online[2].

## 8.2 Results

This section presents the results from each domain that GENPLAN and the four test planners are applied to. Descriptions of the domains are given in Chapter 6. For each domain there is a description of how problems are generated and if any pre-existing benchmark problems are used. The results of running GENPLAN and the other planners on the problem sets are commented upon. The results are presented in order of domain difficulty. Each domain also includes an evaluation of scaling behaviour for efficiency and quality.

### 8.2.1 Maxones

The goal of this domain is to devise a plan that turns all the switches in the problem from off to on. This is an easy domain in which to vary problem difficulty as there is only one variable; the number of switches. Increasing the number of switches increases the difficulty of the problem. The problems were chosen in the range of 5 switches up to 120 switches and were selected in increments of 5. The upper bound of 120 switches

[2] http://homepages.inf.ed.ac.uk/s9638800/index.html

was found through experimentation. All the problems were created deterministically using my own problem generator. This can done because each problem is initialised to have all the switches in the initial state to be off and the goal is to have all switches to be on. Varying the number of switches already on in the initial state would only serve to make the problem easier, therefore the sub-class of having all switches off in the initial state is investigated.

- Let $S =$ the number of switches
  - number of `switch(?s)` actions $= S$
- Action Space: 5-120 actions

#### 8.2.1.1 Results

Referring to Figure 8.1, GENPLAN presents initially comparable scaling then once the problem size increases above 75 switches the scaling becomes poor. The four test planners each present good scaling behaviour for this domain, with BB presenting the best raw efficiency results. There was one technical problem in that IPP was able to solve all problems once an internal variable, "ARRAY_SIZE" had been increased to 120.

In terms of complexity, optimal domain dependent planning for this domain can be achieved with a linear algorithm that runs over a list of switches turning each switch on. This domain should be quite easy according to the planning difficulty metrics: one action bench size, no local minima, and no dead-ends or resources.

Included in Figure 8.1 are the quality results. As expected the returned plan length increases with exactly the number of switches for all five planners. In the case of GENPLAN, this is because once a switch is switched on it cannot be switched off and any action that attempts this is deemed non-working and stripped out of the final plan.

Also included in Figure 8.1 is a result where the maximum plan length parameter for GENPLAN was increased from 200 to 1600 actions and the initial plan length was kept at 60 actions. This experiment was performed to show the effect the maximum plan length has on the efficiency of GENPLAN in conjunction with the `maxones` domain. The "GenPlan-1600" results show that good scaling can be achieved by using a longer maximum plan length for this domain. This result suggests that one of the deficiencies of EA-based search is that they do not make efficient use of the population space assigned to them. This search issue is returned to in Chapter 9.

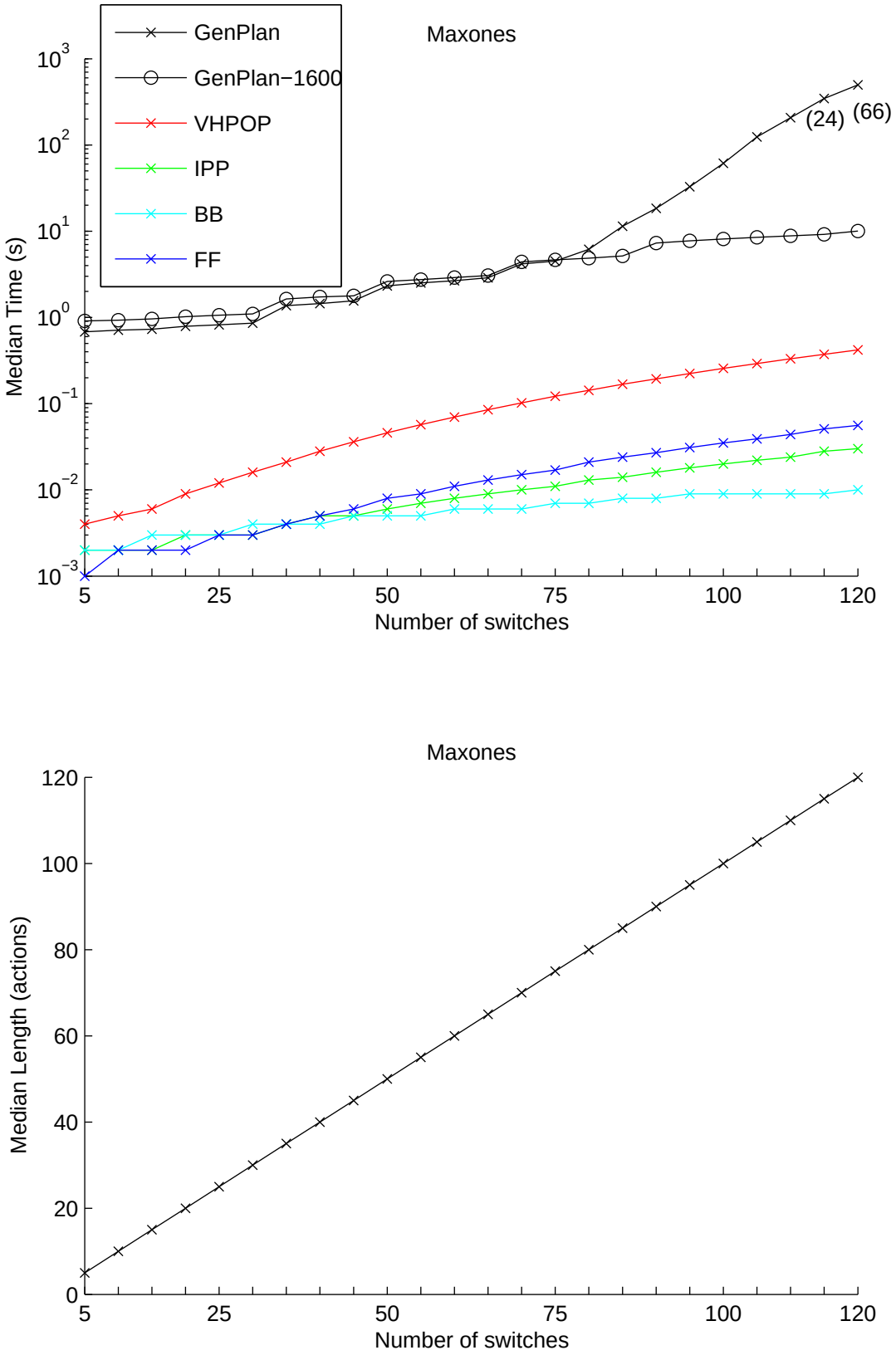



Figure 8.1: Combined efficiency and and plan quality results for `maxones`

#### 8.2.1.2 Evaluation - GenPlan Standard

The results in Figure 8.1 suggest that GENPLAN is scaling more poorly than the other four test planners for this domain.

- **Null Hypothesis** $H_{n0}$: The relationship of solution time between VHPOP and GENPLAN is linear.
  **Alternative Hypothesis**: The relationship of solution time between VHPOP and GENPLAN contains a significant degree of non-linearity.

- **Null Hypothesis** $H_{n1}$: The relationship of solution time between IPP and GENPLAN is linear.
  **Alternative Hypothesis**: The relationship of solution time between IPP and GENPLAN contains a significant degree of non-linearity.

- **Null Hypothesis** $H_{n2}$: The relationship of solution time between BB and GENPLAN is linear.
  **Alternative Hypothesis**: The relationship of solution time between BB and GENPLAN contains a significant degree of non-linearity.

- **Null Hypothesis** $H_{n3}$: The relationship of solution time between FF and GENPLAN is linear.
  **Alternative Hypothesis**: The relationship of solution time between FF and GENPLAN contains a significant degree of non-linearity.

Figure 8.2 visually confirms that GENPLAN is scaling more poorly than the other four test planners for this domain. Table 8.3 presents the relevant statistics related to the regression plot. Each of the four null hypotheses can be rejected because of the p-values, therefore there is a significant degree of non-linearity present in the data. Due to the shape of the resulting curves the four test planners are scaling better than GENPLAN.

#### 8.2.1.3 Evaluation - GenPlan 1600

The results in Figure 8.1 suggest that GENPLAN, with the maximum plan length set to 1600, is scaling better than three of the test planners: VHPOP, FF, and IPP due to the steeper gradient of the lines. In the case of GENPLAN and BB it is more difficult to judge.

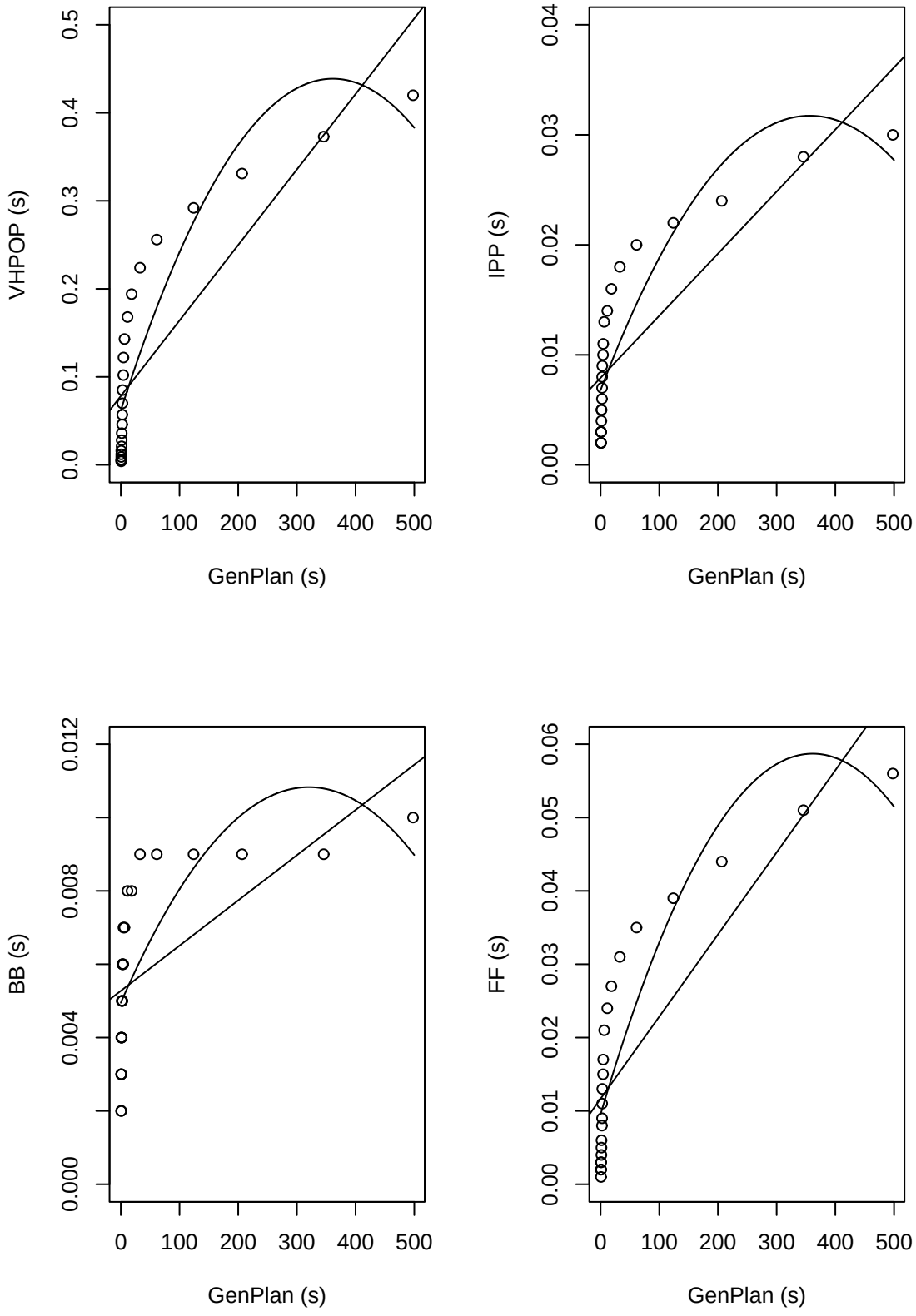



Figure 8.2: Efficiency Regression Plots for `maxones`

| Model | $R^2$ Linear | $R^2$ Quadratic | P-value |
|---|---|---|---|
| $VHPOP \sim GenPlan$ | 0.6945 | 0.8157 | 0.00128 |
| $IPP \sim GenPlan$ | 0.6613 | 0.7843 | 0.00234 |
| $BB \sim GenPlan$ | 0.3744 | 0.4988 | 0.03295 |
| $FF \sim GenPlan$ | 0.6868 | 0.8063 | 0.00169 |

Table 8.3: Statistical data for Figure 8.2

- **Null Hypothesis** $H_{n0}$:The relationship of solution time between VHPOP and GENPLAN is linear.
  **Alternative Hypothesis**: The relationship of solution time between VHPOP and GENPLAN contains a significant degree of non-linearity.

- **Null Hypothesis** $H_{n1}$: The relationship of solution time between IPP and GENPLAN is linear.
  **Alternative Hypothesis**: The relationship of solution time between IPP and GENPLAN contains a significant degree of non-linearity.

- **Null Hypothesis** $H_{n2}$: The relationship of solution time between BB and GENPLAN is linear.
  **Alternative Hypothesis**: The relationship of solution time between BB and GENPLAN contains a significant degree of non-linearity.

- **Null Hypothesis** $H_{n3}$: The relationship of solution time between FF and GENPLAN is linear.
  **Alternative Hypothesis**: The relationship of solution time between FF and GENPLAN contains a significant degree of non-linearity.

Figure 8.3 visually confirms that GENPLAN is scaling better than VHPOP, FF, and IPP, but worse than BB. Table 8.4 presents the relevant statistics related to the regression plot. Each of the four null hypotheses can be rejected confirming a significant degree of non-linearity in the data. Due to the shape of the resulting curves GENPLAN is scaling better than VHPOP, FF, and IPP, and scaling worse than BB.

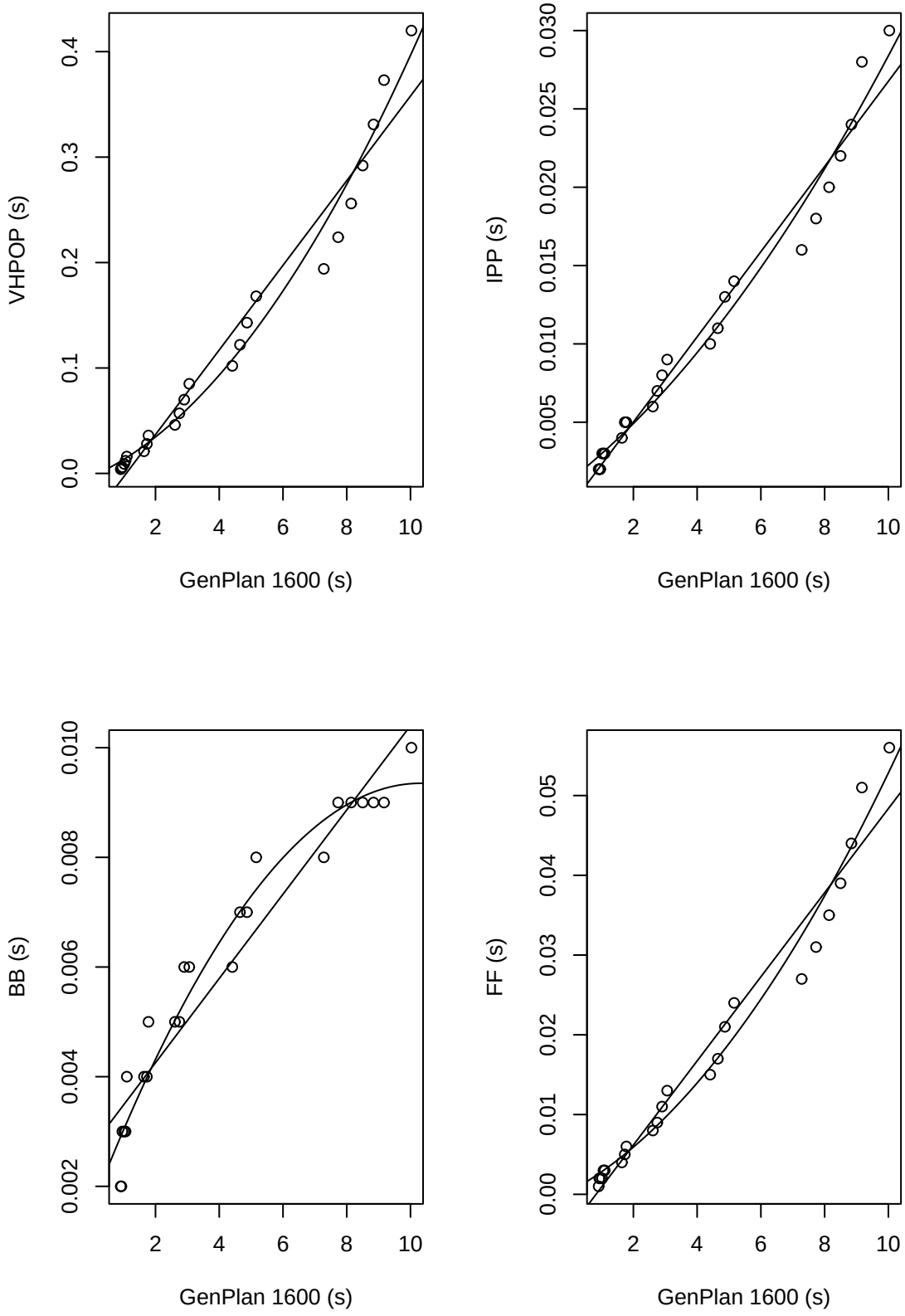



Figure 8.3: Efficiency Regression Plots for `maxones` with GENPLAN's MPL equal to 1600

| Model | $R^2$ Linear | $R^2$ Quadratic | P-value |
|---|---|---|---|
| $VHPOP \sim GenPlan$ | 0.9605 | 0.9805 | 0.000146 |
| $IPP \sim GenPlan$ | 0.9699 | 0.9777 | 0.01323 |
| $BB \sim GenPlan$ | 0.9189 | 0.9579 | 0.000241 |
| $FF \sim GenPlan$ | 0.9625 | 0.9780 | 0.000946 |

Table 8.4: Statistical data for Figure 8.3

### 8.2.2 Travelling Tourist Domain (TTD)

The goal of this domain is to visit all the attractions located at different cities. For this domain there are two variables to vary: the number of cities and the number of attractions. In the initial state all the attractions have not yet been visited. The goal state requires that all attractions are visited. There is no goal for being in any particular city, thus making the local minima value for these problems 0. No attractions start as visited in the initial state as attractions cannot be unvisited. Finally, all attractions are mentioned in the goal state. Attractions are randomly associated with each city so that a city can have no attractions or all the attractions specified at its location. The starting location of the agent was chosen randomly and the problems were generated using my own random problem generator.

The number of cities was varied from 10 to 40 cities in increments of 10. For each specific number of cities, the number of attractions was varied from 2 up to a maximum of 10.

- Let $C$ be the number of cities, and $A$ the number of attractions.
  - number of `go(?c1 ?c2)` actions $= C*(C-1)^3$
  - number of `visit(?c ?a)` actions $= C*A$
- Action Space: 110-1840 actions

#### 8.2.2.1 Results

Referring to Figure 8.4, FF presents the best raw efficiency results and good scaling behaviour for this domain and VHPOP performs almost equally as well. IPP presents

[3] Actions attempting to go to the city you are currently in are removed. This is true for any domain involving locations in this chapter.

a mixture of good and poor scaling behaviour depending on the problem set. Initially BB's raw performance is much better than that of GENPLAN's but as problem difficulty increases they exhibit almost equally poor efficiency and poor scaling. Generally, GENPLAN appears to be scaling worse than the other test planners except potentially BB and IPP. In addition, in order to force BB to solve additional problems the memory bound was increased to 2Gb. Even with this additional memory the planner was unable to solve the 40 cities and 7 attractions problem.

The main difference between this domain and `maxones` is that it typically takes two actions to achieve a goal, though it can take a single action if there is more than one attraction in a city where the agent is currently located or if the agent's initial location contains an attraction. In addition, the problems have much larger action spaces, in fact the largest of any of the domains that will be considered. The increase in bench size and action space over `maxones` has affected the efficiency of GENPLAN, BB, and IPP. Optimal planning for this domain is polynomial as a domain dependent algorithm can simply look up the locations of all attractions and visit them in any order visiting all attractions at each location. This domain does not feature local minima, dead-ends or resources and so it should be relatively easy.

Referring to Figure 8.5 all planers present linearly increasing plan lengths for the differing problem sizes. The four test planners all returned the same length of plan for all problems. GENPLAN is returning plan lengths which are a constant size longer than those plans produced by the four test planners.

#### 8.2.2.2 Evaluation - Efficiency

The previous section speculated that the scaling behaviour of GENPLAN is equivalent to that of BB and possibly IPP and that VHPOP and FF scale better than GENPLAN. These intuitions will be investigated in terms of specific problems sets in order to avoid dominance and grouping effects. For each comparison with GENPLAN and another planner there are four separate hypotheses depending on which problem size is being considered as indicated by the set notation.

- **Null Hypothesis**: The relationship of solution time between VHPOP and GENPLAN is linear for the $\{10,20,30,40\}$ city problems.
  **Alternative Hypothesis**: The relationship of solution time between VHPOP and GENPLAN contains a significant degree of non-linearity for the $\{10,20,30,40\}$ city problems.

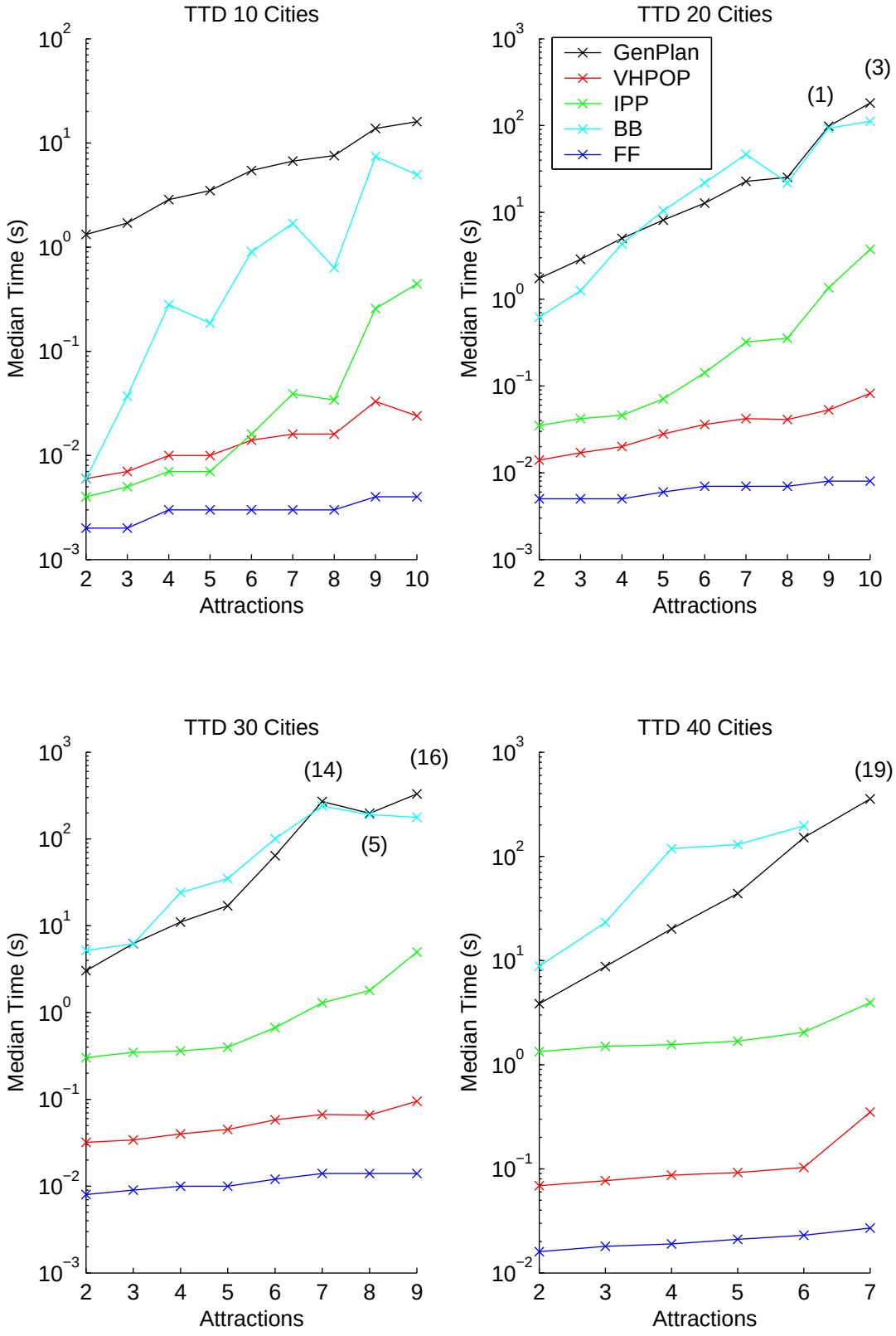



Figure 8.4: Efficiency results for TTD

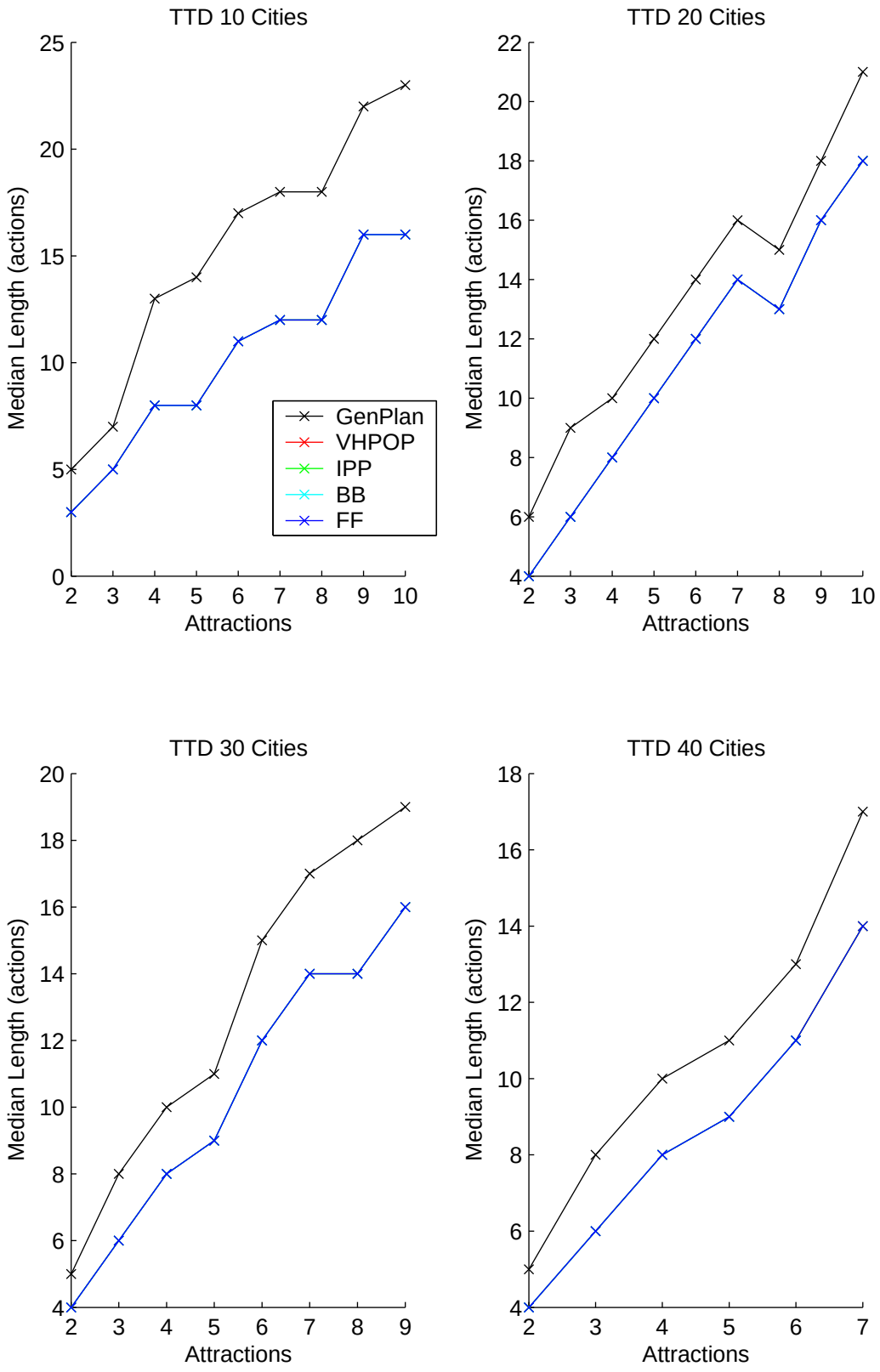



Figure 8.5: Plan quality results for TTD

| Domain | VHPOP | IPP | BB | FF |
|---|---|---|---|---|
| TTD 10 Cities | inc. | better | inc. | inc. |
| TTD 20 Cities | inc. | better | inc. | worse |
| TTD 30 Cities | inc. | inc. | worse | worse |
| TTD 40 Cities | better | better | inc. | inc. |

Table 8.5: A summary of the hypotheses based on Table 8.6 and Figure 8.6

- **Null Hypothesis**: The relationship of solution time between IPP and GENPLAN is linear for the $\{10,20,30,40\}$ city problems.
  **Alternative Hypothesis**: The relationship of solution time between IPP and GENPLAN contains a significant degree of non-linearity for the $\{10,20,30,40\}$ city problems.

- **Null Hypothesis**: The relationship of solution time between BB and GENPLAN is linear for the $\{10,20,30,40\}$ city problems.
  **Alternative Hypothesis** The relationship of solution time between BB and GENPLAN contains a significant degree of non-linearity for the $\{10,20,30,40\}$ city problems.

- **Null Hypothesis**: The relationship of solution time between FF and GENPLAN is linear for the $\{10,20,30,40\}$ city problems.
  **Alternative Hypothesis** The relationship of solution time between FF and GENPLAN contains a significant degree of non-linearity for the $\{10,20,30,40\}$ city problems.

The discussion refers to Table 8.6, Figure 8.6, and Table 8.5. A "better" means that there was sufficient evidence to reject the null hypothesis and due to the shape of the curve GENPLAN is scaling better than another planner, "inc." means that there was insufficient evident to reject the null hypothesis, "worse" means that there was sufficient evidence to reject the null hypothesis and due to the shape of the curve GENPLAN is scaling worse than another planner.

The test was unable to support the conclusion that VHPOP is scaling better than GENPLAN. Surprisingly, the test supports better scaling for GENPLAN than VHPOP on the 40 City problems. This is caused by the kink for the 40-7 problem seen in Figure

| Model - Efficiency | $R^2$ Linear | $R^2$ Quadratic | P-value |
|---|---|---|---|
| $VHPOP \sim GenPlan$ (10 cities) | 0.8644 | 0.8855 | 0.3343 |
| $VHPOP \sim GenPlan$ (20 cities) | 0.8705 | 0.8857 | 0.4064 |
| $VHPOP \sim GenPlan$ (30 cities) | 0.8784 | 0.8788 | 0.9099 |
| $VHPOP \sim GenPlan$ (40 cities) | 0.9029 | 0.9915 | 0.01133 |
| $IPP \sim GenPlan$ (10 cities) | 0.8568 | 0.9918 | $<$0.0001 |
| $IPP \sim GenPlan$ (20 cities) | 0.9730 | 0.9980 | 0.00013 |
| $IPP \sim GenPlan$ (30 cities) | 0.7241 | 0.8387 | 0.118 |
| $IPP \sim GenPlan$ (40 cities) | 0.9678 | 0.9931 | 0.0451 |
| $BB \sim GenPlan$ (10 cities) | 0.8144 | 0.8254 | 0.562 |
| $BB \sim GenPlan$ (20 cities) | 0.8944 | 0.9662 | 0.0118 |
| $BB \sim GenPlan$ (30 cities) | 0.8544 | 0.9814 | 0.002075 |
| $BB \sim GenPlan$ (40 cities) | 0.7222 | 0.9071 | 0.1840 |
| $FF \sim GenPlan$ (10 cities) | 0.8219 | 0.8472 | 0.3570 |
| $FF \sim GenPlan$ (20 cities) | 0.5604 | 0.7852 | 0.0461 |
| $FF \sim GenPlan$ (30 cities) | 0.8225 | 0.9477 | 0.01805 |
| $FF \sim GenPlan$ (40 cities) | 0.8814 | 0.9296 | 0.2469 |

Table 8.6: Statistical data for Figure 8.6

8.4. Further sampling is required to make this conclusion more robust. The comparison between GENPLAN and IPP over specific data sets has revealed that GENPLAN can scale better than another planner. The testing between GENPLAN and BB is inconclusive, except for the 30 city result, which is probably due to the noisy data present in the BB curves. Finally, the test showed that FF is scaling better than GENPLAN for the 20 and 30 city problem sets.

#### 8.2.2.3 Evaluation - Plan Quality

The results in Figure 8.7 includes all the data present in Figure 8.5. Figure 8.7 supports the conclusion that the scaling behaviour of GENPLAN for plan quality is equivalent to each of the four test planners. Referring to Table 8.7, the $R^2$ values for each linear model is high which indicates a strong linear relationship between the quality of plans produced by GENPLAN and those of the four test planners. In addition, the non-linearity test reported that there is insufficient evidence to claim a significant non-

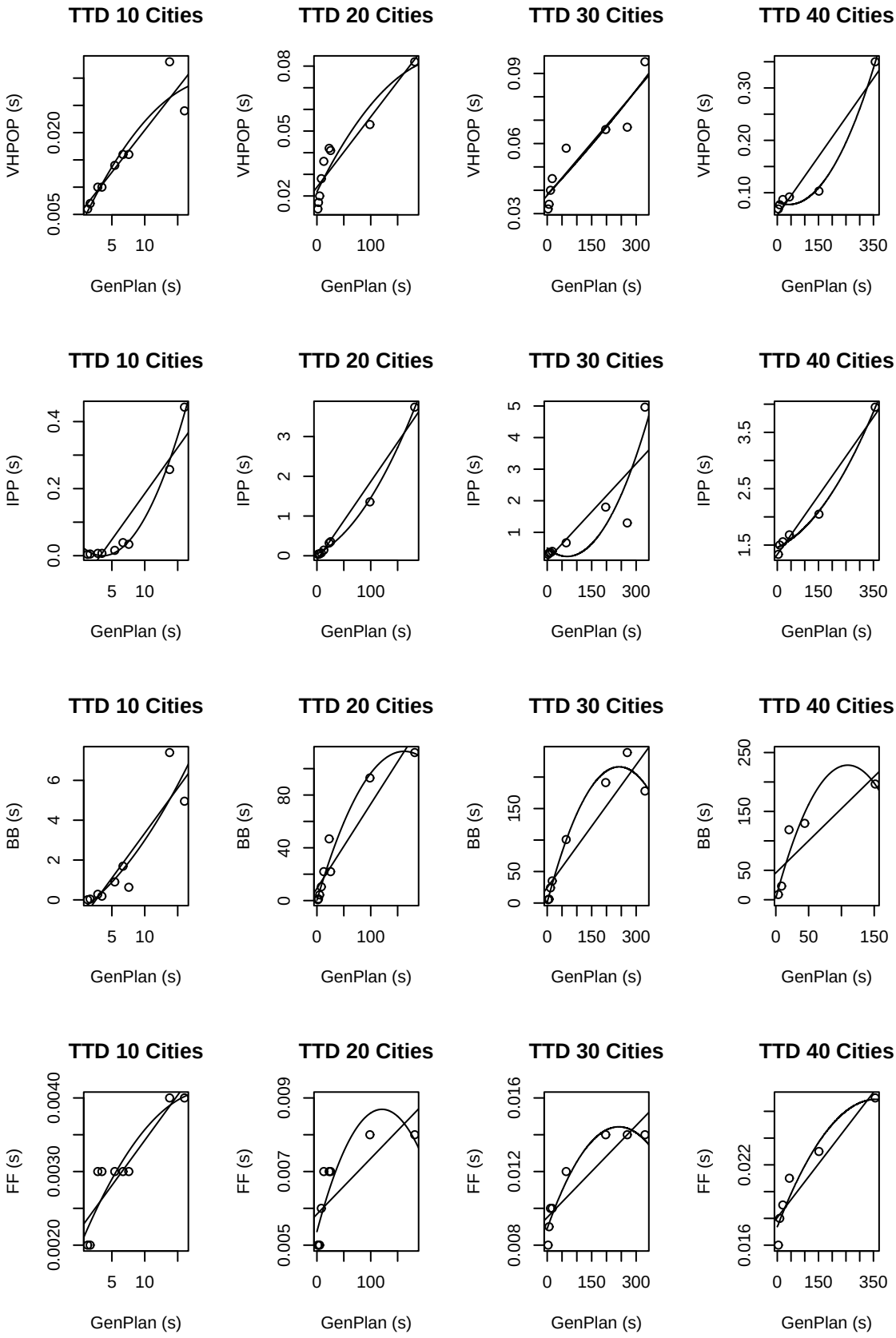



Figure 8.6: Regression plots for efficiency for TTD

| Model - Quality | $R^2$ Linear |
|---|---|
| $VHPOP \sim GenPlan$ | 0.9099 |
| $IPP \sim GenPlan$ | 0.9099 |
| $BB \sim GenPlan$ | 0.9090 |
| $FF \sim GenPlan$ | 0.9099 |

Table 8.7: Statistical data for Figure 8.7

linearity in the data for each comparison between GENPLAN and another test planner.

### 8.2.3 Briefcase

The goal of this domain is to transport objects between locations using a briefcase. For this domain there are two variables to vary. The first being the number of locations and second the number of objects. For these experiments the number of briefcases was fixed to one. Extra briefcases would add an extra layer of difficulty that would obscure the examination of efficiency of planning for this domain. As can been seen from the action space calculation below, extra briefcases will increase the action space, creating much harder problems, and this issue is returned to in Chapter 9. Resources are not ignored as both `driverlog` and `satellite` have multiple usable resources. The goal state never specifies a goal location for the briefcase, therefore the size of local minima for this class of problems is 0. The objects never start in their goal location as this would only serve to make problems easier and the main problem of the domain is to transport objects. Any location can have any number of objects to start off with and the briefcase can start at any location. The goal state specifies goal locations for all objects in the problem.

Problems sizes were restricted to 6, 7, 8, and 9 cities. For each specific number of cities, the number of objects was varied from 2 up to 7 objects. None of the instances in the previous chapter were used and these are all new instances. The problems were created using my own random problem generator.

- Let $O$ be the number of objects, let $L$ be the number of locations, and let $B$ be the number of briefcases.
  - number of `putin (?o ?b ?l)` action $= O * B * L$

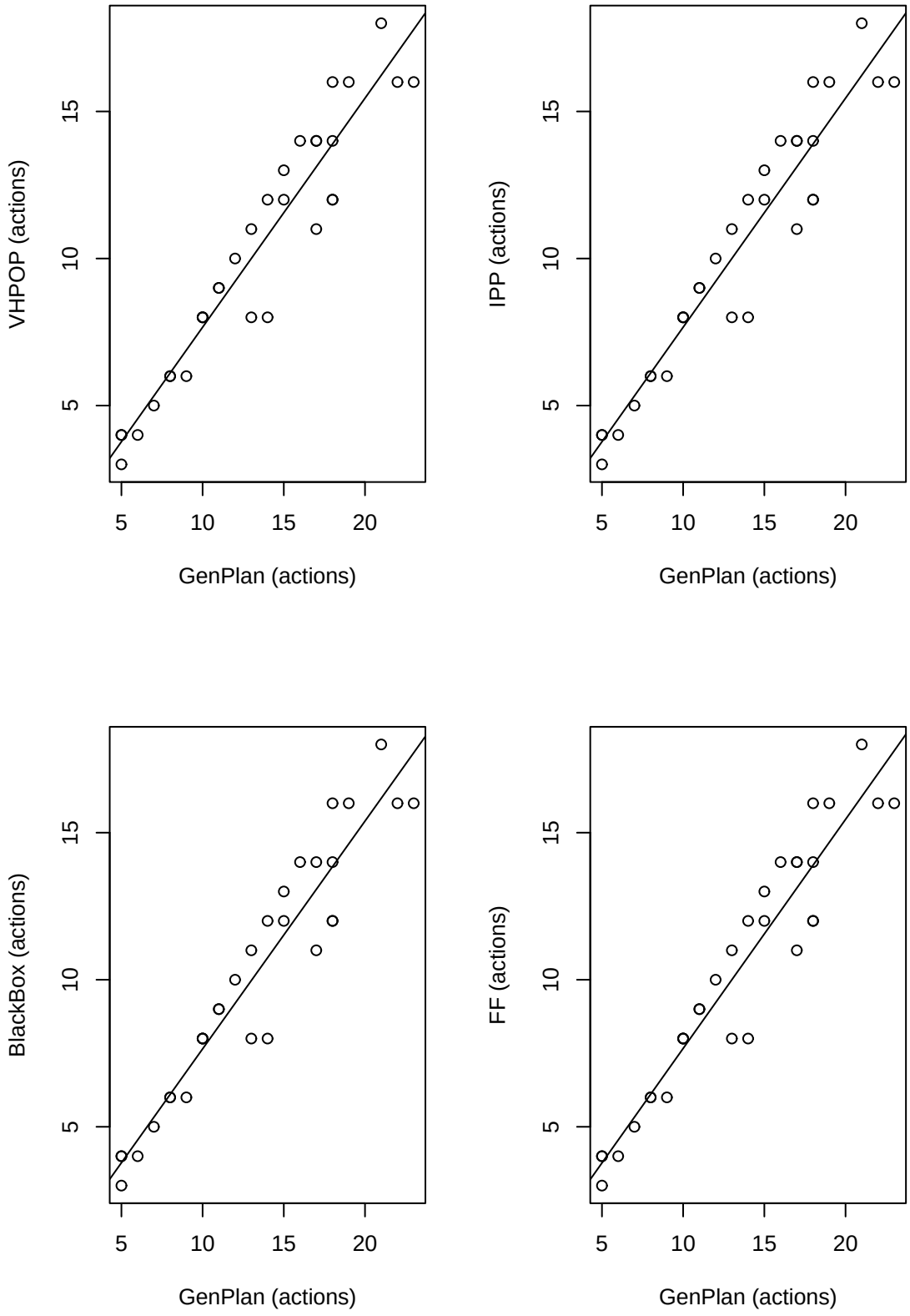



Figure 8.7: Regression plots for plan quality for TTD

  - number of `takeout(?o ?b ?l)` actions $= O * B * L$
  - number of `movebriefcase(?b ?l1 ?l2)` actions $= B * L * (L-1)$

- Action Space: 54-198 actions

#### 8.2.3.1 Results

Referring to Figure 8.8, GENPLAN, IPP, and BB all present poor scaling for this domain. The scaling for IPP appears worse than that of GENPLAN due to the curvature of line. The scaling behaviour of GENPLAN as compared with BB appears equivalent. VHPOP and FF present good scaling on this domain, though VHPOP does have more variable efficiency. FF presents the best raw efficiency results.

The number of cities has been reduced further to accommodate the increasing complexity of the planning domain. The domain is more complex as it now takes at least 3 actions to achieve a goal and the domain involves an extra operator when compared to `TTD`. The problems involve smaller action spaces than compared with `TTD`. There are no dead-ends in this domain, or local minima for the selected problems. This domain also makes use of a single infinite carrying resource, the next domain, `gripper`, has a restricted carrying resource.

Referring to Figure 8.9, all planers show constantly increasing plan length for the differing problem sets. The four test planners all return roughly the same length of plan for the four sets of problems. GENPLAN is returning plan lengths which are a constant size longer then those plans produced by the test planners.

#### 8.2.3.2 Evaluation - Efficiency

The results in Figure 8.8 suggest that the scaling behaviour of GENPLAN is equivalent to that of BB, better than IPP, and that VHPOP and FF scale better than GENPLAN. These intuitions will be investigated in terms of specific problems sets in order to avoid dominance and grouping effects. For each comparison with GENPLAN and another planner there are four separate hypotheses depending on which problem size is being considered as indicated by the set notation.

- **Null Hypothesis**: The relationship of solution time between VHPOP and GENPLAN is linear for the $\{6,7,8,9\}$ city problems.
  **Alternative Hypothesis**: The relationship of solution time between VHPOP and

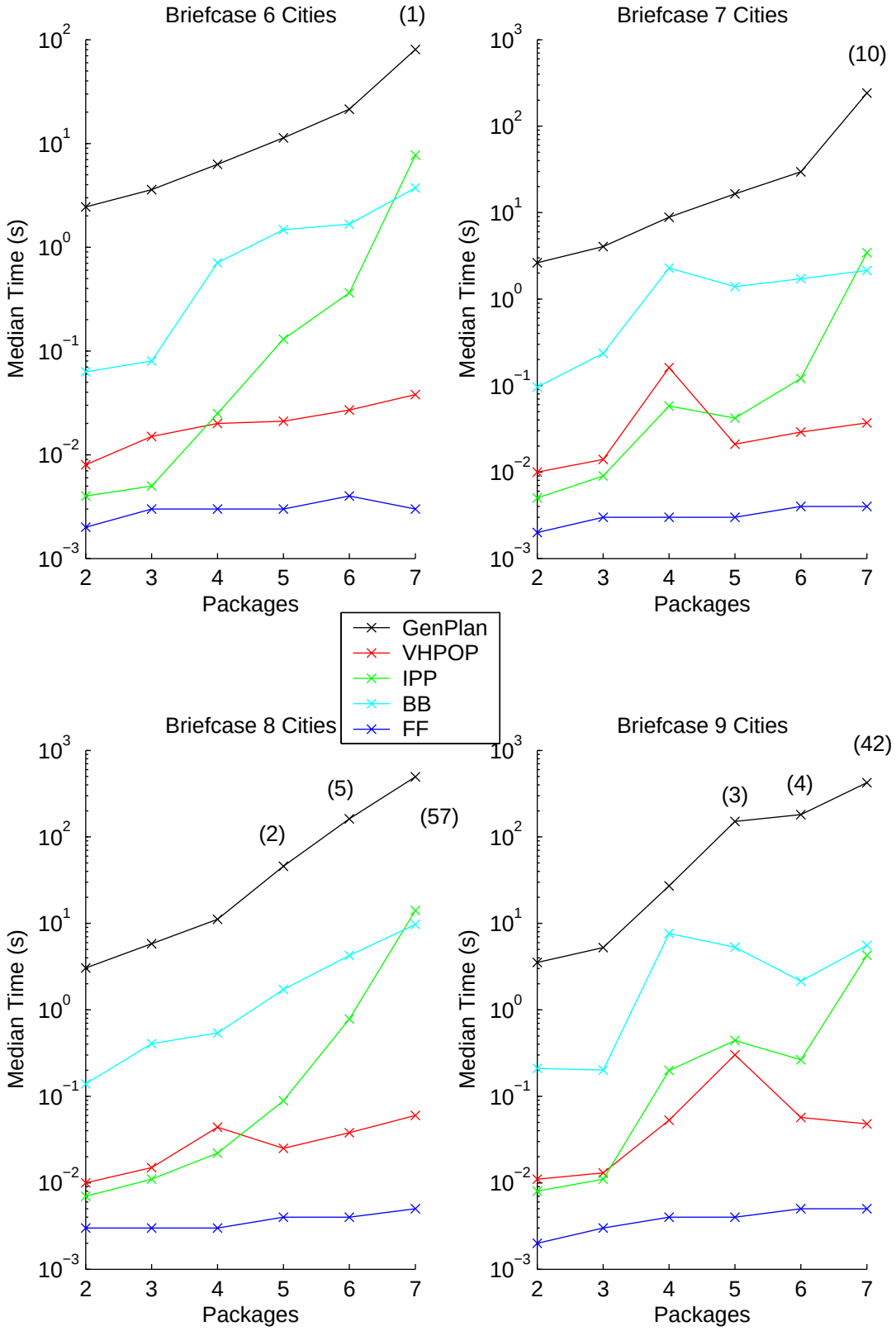



Figure 8.8: Efficiency results for `briefcase`

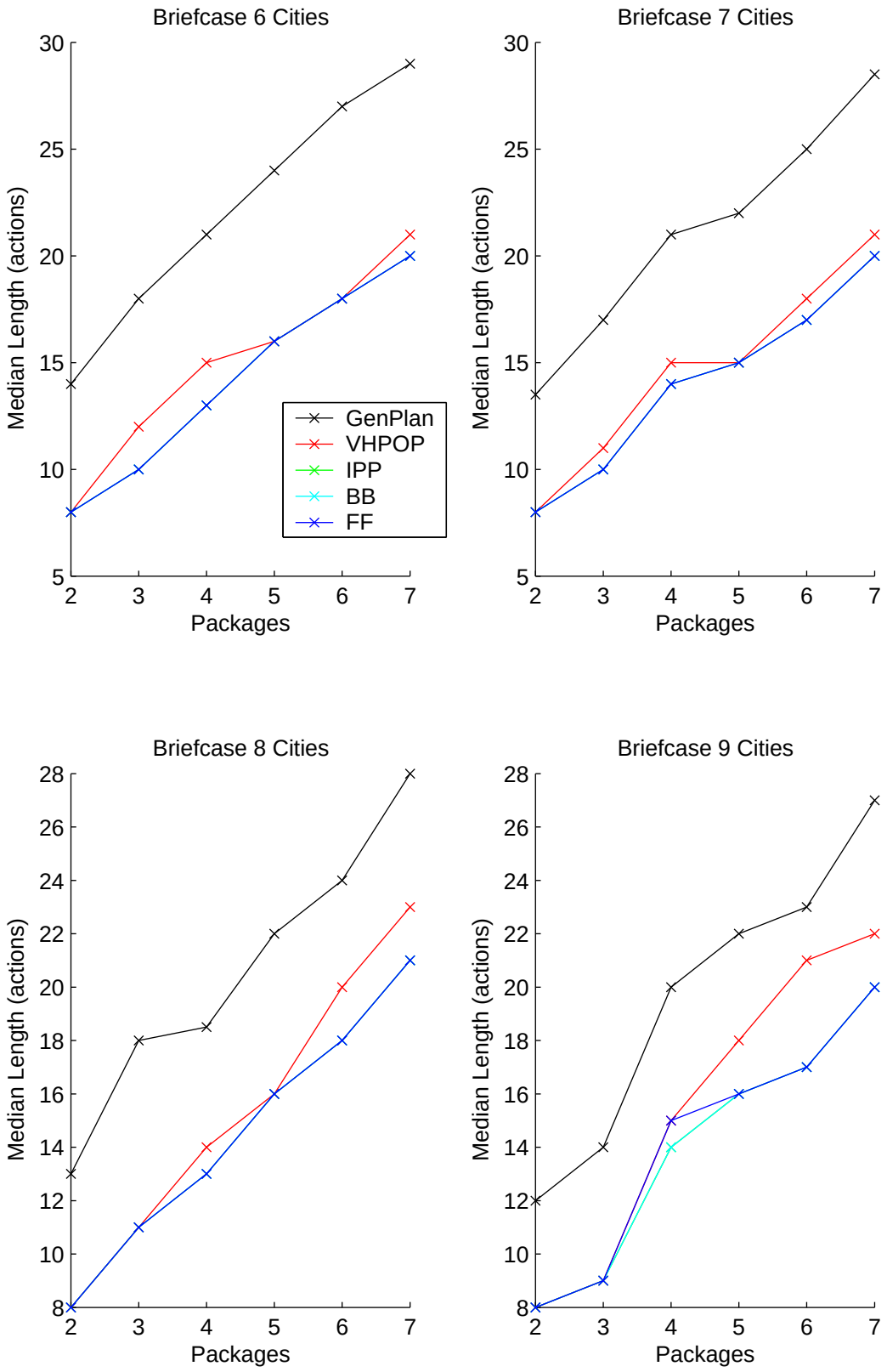



Figure 8.9: Plan quality results for `briefcase`

GENPLAN contains a significant degree of non-linearity for the $\{6,7,8,9\}$ city problems.

- **Null Hypothesis**: The relationship of solution time between IPP and GENPLAN is linear for the $\{6,7,8,9\}$ city problems.
  **Alternative Hypothesis**: The relationship of solution time between IPP and GENPLAN contains a significant degree of non-linearity for the $\{6,7,8,9\}$ city problems.

- **Null Hypothesis**: The relationship of solution time between BB and GENPLAN is linear for the $\{6,7,8,9\}$ city problems.
  **Alternative Hypothesis** The relationship of solution time between BB and GENPLAN contains a significant degree of non-linearity for the $\{6,7,8,9\}$ city problems.

- **Null Hypothesis**: The relationship of solution time between FF and GENPLAN is linear for the $\{6,7,8,9\}$ city problems.
  **Alternative Hypothesis** The relationship of solution time between FF and GENPLAN contains a significant degree of non-linearity for the $\{6,7,8,9\}$ city problems.

The discussion refers to Table 8.8, Table 8.9, and Figure 8.10. Though the scaling behaviour of GENPLAN appears worse than that of VHPOP in Figure 8.8 this is not confirmed by the test for any of the problem sets. The comparison between GENPLAN and IPP over specific data sets has revealed that GENPLAN can scale better than IPP for all the problem sets. The comparison between GENPLAN and BB resulted in one significant result for the 8 City problems reporting that GENPLAN is scaling worse than BB for this problem set. Though the scaling behaviour of GENPLAN appears worse than that of FF in Figure 8.8 this is also not confirmed by any of the problem sets.

#### 8.2.3.3 Evaluation - Plan Quality

Figure 8.11 includes all the data present in Figure 8.9. Figure 8.11 supports the conclusion that the scaling behaviour of GENPLAN for plan quality is equivalent to each of the four test planners. Referring to Table 8.10, the $R^2$ values for the each linear model

| Domain | VHPOP | IPP | BB | FF |
|---|---|---|---|---|
| Briefcase 6 Cities | inc. | better | inc. | inc. |
| Briefcase 7 Cities | inc. | better | inc. | inc. |
| Briefcase 8 Cities | inc. | better | worse | inc. |
| Briefcase 9 Cities | inc. | better | inc. | inc. |

Table 8.8: A summary of the hypotheses based on Table 8.9 and Figure 8.10

| Model - Efficiency | $R^2$ Linear | $R^2$ Non-linear | P-value |
|---|---|---|---|
| $VHPOP \sim GenPlan$ 6 Cities | 0.7932 | 0.9277 | 0.0992 |
| $VHPOP \sim GenPlan$ 7 Cities | 0.0059 | 0.007827 | 0.945 |
| $VHPOP \sim GenPlan$ 8 Cities | 0.6292 | 0.6391 | 0.794 |
| $VHPOP \sim GenPlan$ 9 Cities | 0.02436 | 0.4546 | 0.222 |
| $IPP \sim GenPlan$ 6 Cities | 0.9647 | 0.9999 | $<$0.0001 |
| $IPP \sim GenPlan$ 7 Cities | 0.9940 | 0.9999 | 0.00114 |
| $IPP \sim GenPlan$ 8 Cities | 0.9325 | 0.9998 | $<$0.0001 |
| $IPP \sim GenPlan$ 9 Cities | 0.8383 | 0.9917 | 0.00497 |
| $BB \sim GenPlan$ 6 Cities | 0.9038 | 0.9668 | 0.0970 |
| $BB \sim GenPlan$ 7 Cities | 0.2358 | 0.4787 | 0.322 |
| $BB \sim GenPlan$ 8 Cities | 0.9870 | 0.9987 | 0.0139 |
| $BB \sim GenPlan$ 9 Cities | 0.1515 | 0.1645 | 0.8430 |
| $FF \sim GenPlan$ 6 Cities | 0.03987 | 0.7541 | 0.05995 |
| $FF \sim GenPlan$ 7 Cities | 0.3787 | 0.8175 | 0.07466 |
| $FF \sim GenPlan$ 8 Cities | 0.8240 | 0.8788 | 0.3286 |
| $FF \sim GenPlan$ 9 Cities | 0.5843 | 0.7457 | 0.2615 |

Table 8.9: Statistical data for Figure 8.10

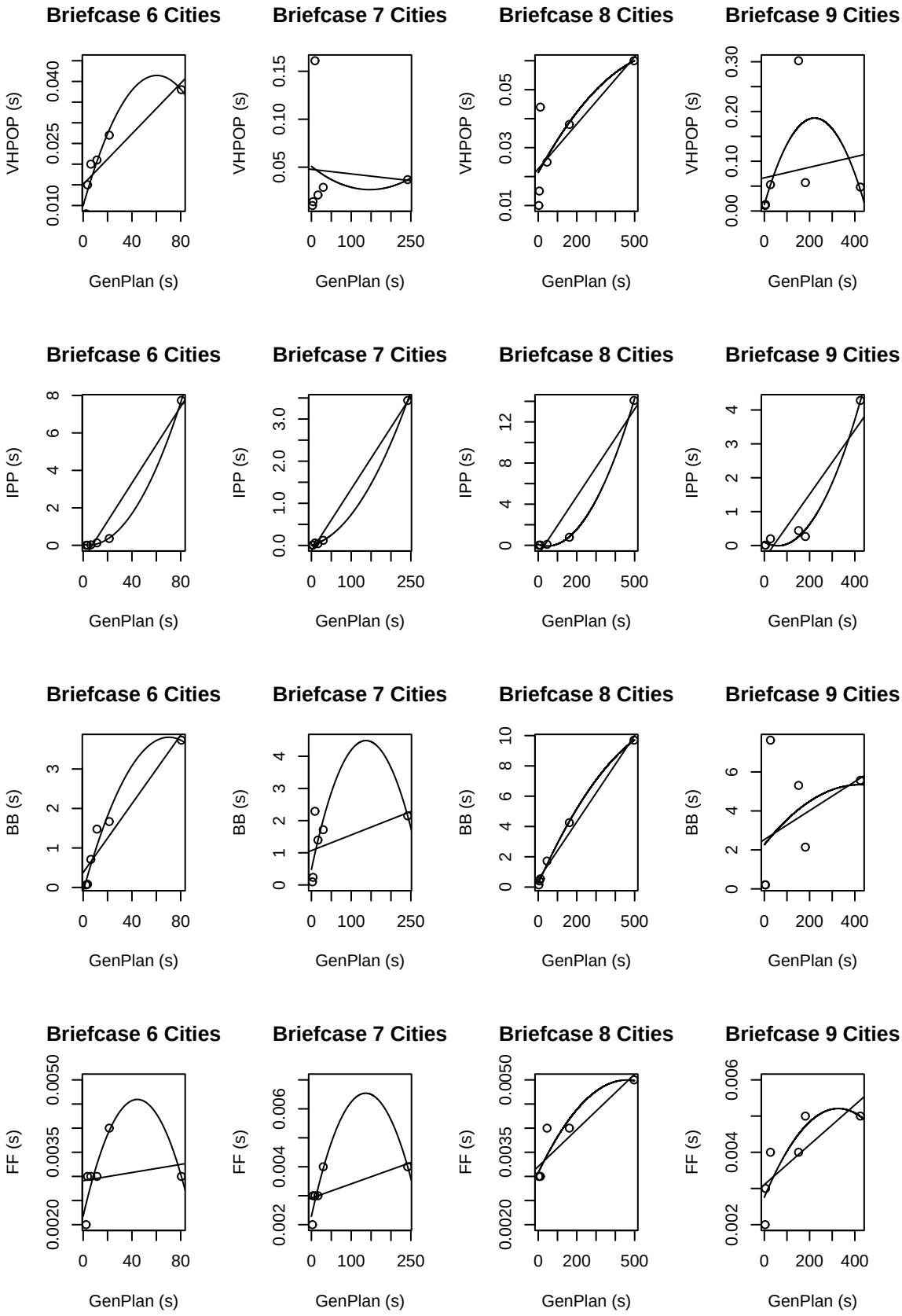



Figure 8.10: Regression plots for efficiency for `briefcase`

| Model - Quality | $R^2$ Linear |
|---|---|
| $VHPOP \sim GenPlan$ | 0.9095 |
| $IPP \sim GenPlan$ | 0.9570 |
| $BB \sim GenPlan$ | 0.9570 |
| $FF \sim GenPlan$ | 0.9523 |

Table 8.10: Statistical data for Figure 8.11

is high which indicates a strong linear relationship between the quality of plans produced by GENPLAN and those of the four test planners. In addition, the non-linearity test reported that there is insufficient evidence to claim a significant non-linearity in the data for each comparison between GENPLAN and another test planner.

### 8.2.4 Gripper

The goal of this domain is to transport objects between two locations but by using a carrier that is restricted in how much it can carry. There are two rooms and the robot has two grippers with which to carry balls. All the balls start in room one along with the robot with both its grippers free. The goal state is to have all the balls in room two. The goal state also specifies a goal location for the robot, which is room two, making a one action local minima. The state of the grippers is not specified in the goal state. Like the `briefcase` domain, it takes at least 3 actions to achieve a goal. For this domain there is only one variable to vary, the number of balls, and the problems were specified in this fashion in order to mimic the problems used for IPC 1. A direct comparison between results from IPC 1 and here is not justified as this domain uses typed STRIPS, whereas the original domain is untyped. This allows for a smaller action space as any parameter of an operator is no longer free to take any object, due to the restrictions of typing.

The number of balls is varied from 2 to 14. The problems were generated deterministically due to the restrictions made to the problems using my own generator.

- Let $R$ be the number of rooms, $B$ the number of balls, and $G$ the number of grippers.
  - number of `move(?r1 ?r2)` actions $= R * R - 1$

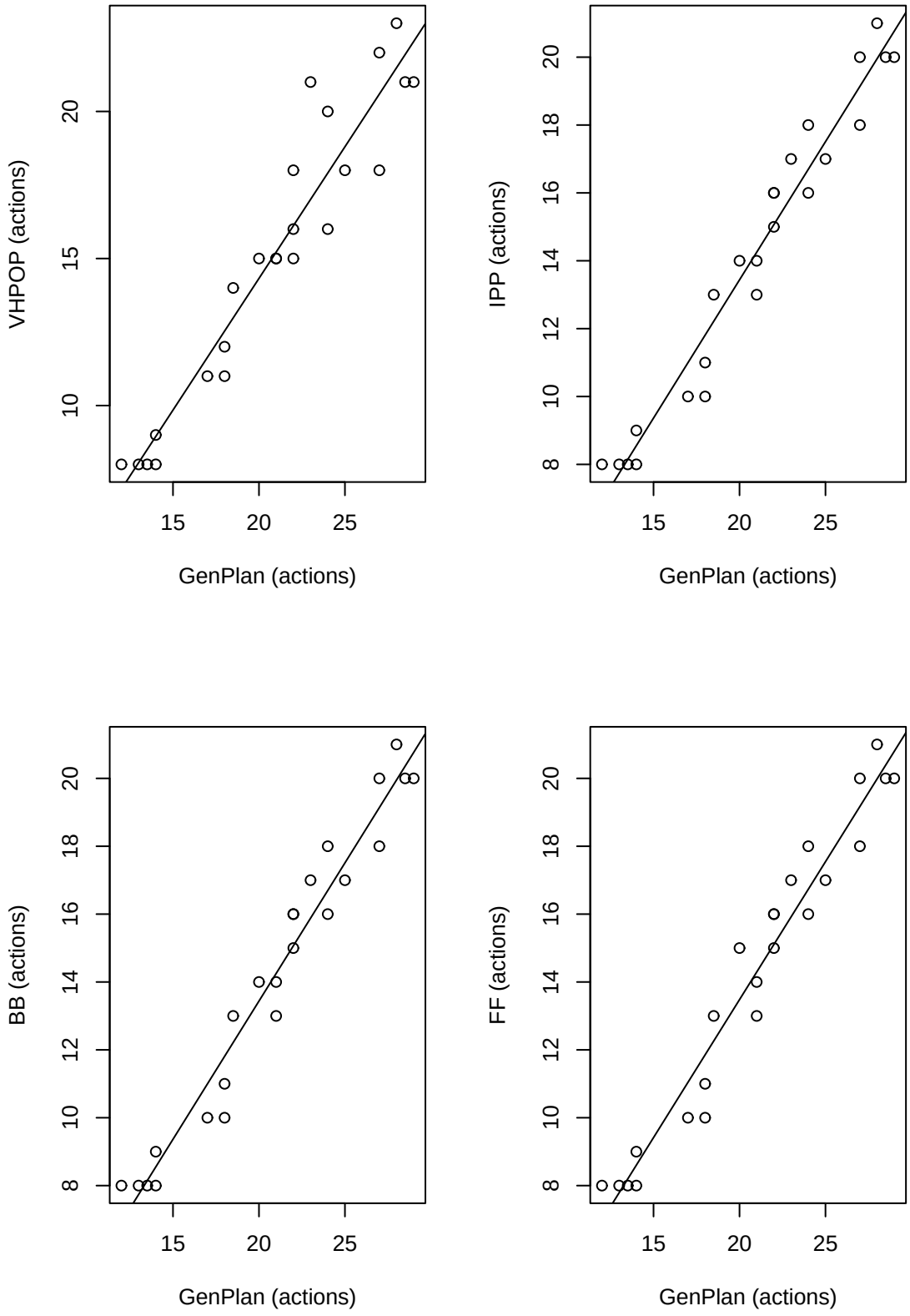



Figure 8.11: Regression plots for plan quality for `briefcase`

- number of `pick(?b ?r ?g)` actions $= B*R*G$
- number of `drop(?b ?r ?g` actions $= B*R*G$

- Action Space: 10-114

#### 8.2.4.1 Results

Referring to Figure 8.12 GENPLAN, VHPOP, IPP, and BB present poor scaling behaviour for the `gripper` domain with either VHPOP or BB scaling the worst. VHPOP exceeded the 2Gb memory limit for the 7 ball problem, IPP exceeded the 2Gb memory limit for the 13 ball problem, and BB exceeded the 500 second time limit for the 7 ball problem so I was unable to force the other planners to solve any additional problems. FF presents the best raw performance and scaling behaviour for this domain.

This domain presents a number of additional issues that have not yet been touched on by the domains examined so far in this chapter. The main increase in difficulty introduced by this domain is that a maximum of only two balls can be carried at any time. The limited carrying resource issue is returned to in Chapter 9 using a comparison with `briefcase` and `gripper`. The effect of having multiple one action local minima will also be examined in Chapter 9.

Referring to Figure 8.12, the second plot, GENPLAN is producing plans that are longer than those produced by the four test planners. In particular, much longer plans in the range of 3 to 8 balls, but after 9 balls the GENPLAN plans appear to be a constant length longer than those produced by FF and IPP. The four test planners all produce plans of equal length.

#### 8.2.4.2 Evaluation - Efficiency

The results in Figure 8.12 suggest that GENPLAN is scaling better VHPOP, IPP, and BB, but worse than FF. The tests are carried out using all efficiency data in Figure 8.12. In the case of the BB and GENPLAN comparison an additional result of 500 seconds was added for BB on the 7 ball problem as it fails to find a plan within the time bound. The following hypotheses will investigate the intuitions:

- **Null Hypothesis** $H_{n0}$: The relationship of solution time between VHPOP and GENPLAN is linear.
  **Alternative Hypothesis**: The relationship of solution time between VHPOP and GENPLAN contains a significant degree of non-linearity.

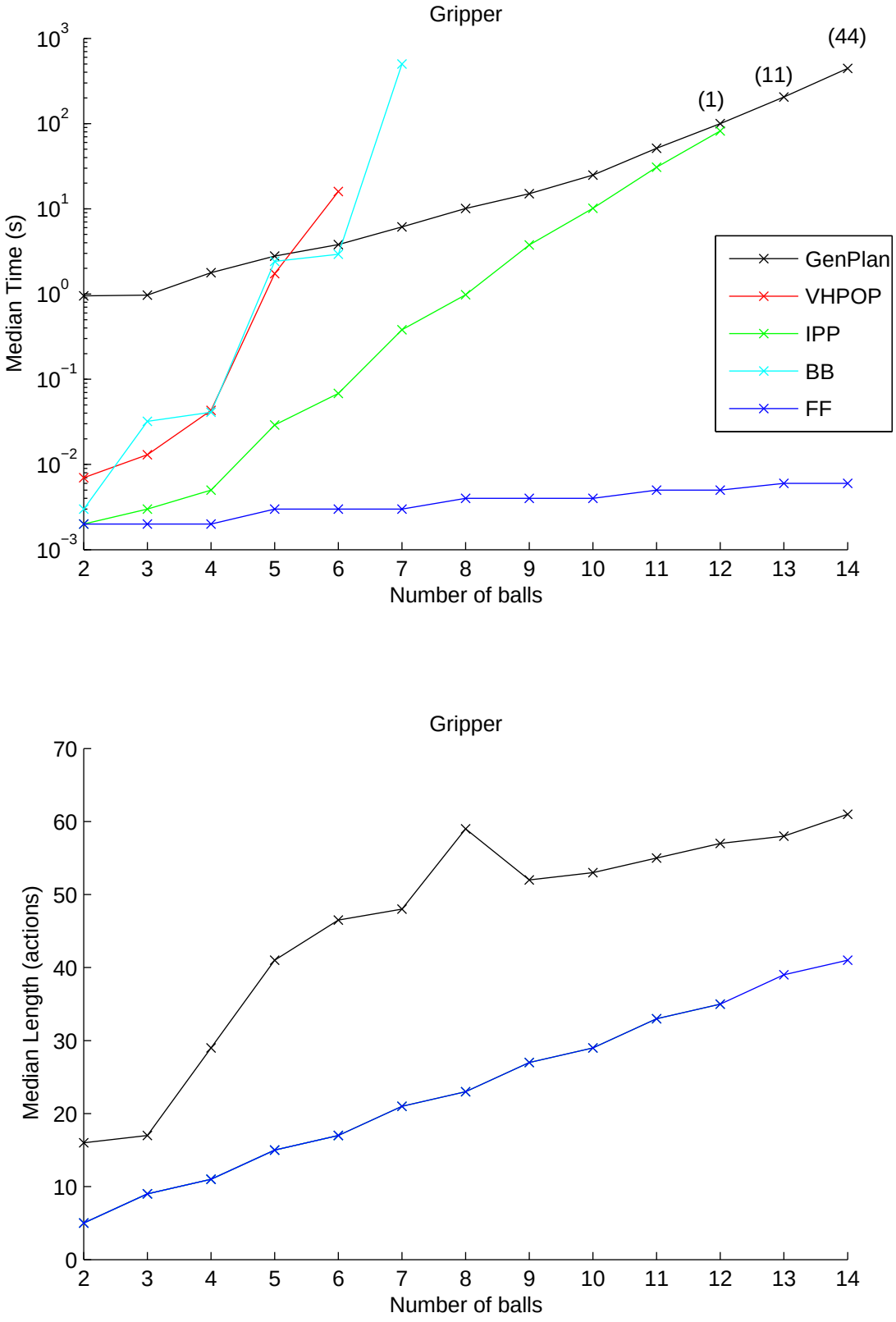



Figure 8.12: Combined efficiency and plan quality quality results for `gripper`

- **Null Hypothesis** $H_{n1}$: The relationship of solution time between IPP and GENPLAN is linear.
  **Alternative Hypothesis**: The relationship of solution time between IPP and GENPLAN contains a significant degree of non-linearity.

- **Null Hypothesis** $H_{n2}$: The relationship of solution time between BB and GENPLAN is linear.
  **Alternative Hypothesis**: The relationship of solution time between BB and GENPLAN contains a significant degree of non-linearity.

- **Null Hypothesis** $H_{n3}$: The relationship of solution time between FF and GENPLAN is linear.
  **Alternative Hypothesis** The relationship of solution time between FF and GENPLAN contains a significant degree of non-linearity.

The discussion refers to Table 8.11 and Figure 8.13:

- There is sufficient evidence to reject $H_{n0}$. There is a significant degree of non-linearity and due to the shape of the curve GENPLAN is scaling better than VHPOP.

- There is sufficient evidence to reject $H_{n1}$. There is a significant degree of non-linearity and due to the shape of the curve GENPLAN is scaling better than IPP.

- There is sufficient evidence to reject $H_{n1}$. There is a significant degree of non-linearity and due to the shape of the curve GENPLAN is scaling better than BB.

- There is sufficient evidence to reject $H_{n3}$. There is a significant degree of non-linearity and due to the shape of the curve GENPLAN is scaling worse than FF.

The test confirmed that GENPLAN is scaling better than VHPOP, IPP, and BB. FF is also confirmed to be scaling better than GENPLAN.

#### 8.2.4.3 Evaluation - Quality

The results in Figure 8.14 includes all the quality results present in Figure 8.12. Figure 8.14 suggests there may be some non-linearity between GENPLAN and the test planners. The non-linearity test reported that there is insufficient evidence to claim a significant non-linearity in the data for each comparison between GENPLAN and another test planner. Referring to Table 8.12, the $R^2$ values for the each linear model are

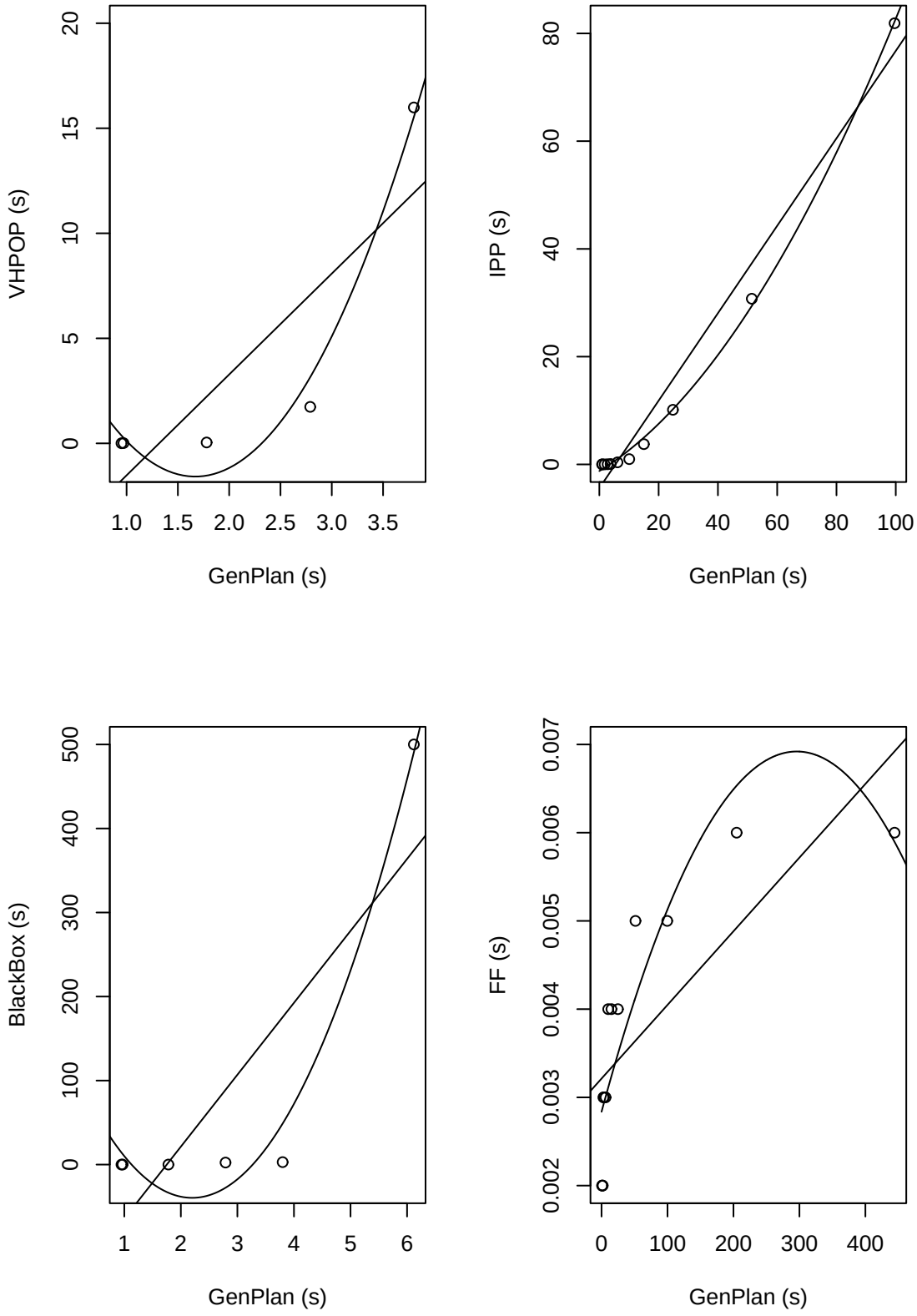



Figure 8.13: Regression plots for efficiency for `gripper`

| Model - Efficiency | $R^2$ Linear | $R^2$ Non-linear | P-value |
|---|---|---|---|
| $VHPOP \sim GenPlan$ | 0.7148 | 0.9749 | 0.045 |
| $IPP \sim GenPlan$ | 0.9729 | 0.9986 | $<$0.0001 |
| $BB \sim GenPlan$ | 0.6997 | 0.9779 | 0.00864 |
| $FF \sim GenPlan$ | 0.5595 | 0.7954 | 0.00682 |

Table 8.11: Statistical data for Figure 8.13

| Model - Quality | $R^2$ Linear |
|---|---|
| $VHPOP \sim GenPlan$ | 0.9259 |
| $IPP \sim GenPlan$ | 0.8025 |
| $BB \sim GenPlan$ | 0.9259 |
| $FF \sim GenPlan$ | 0.7920 |

Table 8.12: Statistical data for Figure 8.14

good which indicates a linear relationship between the quality of plans produced by GENPLAN and those of the four test planners.

### 8.2.5 Blocksworld-3ops

The goal of this domain is to transform one multi-stack of blocks into another multi-stack of blocks. There is one variable to vary in this domain which is the number of blocks. All the problems were created using the random problem generator BW-STATES which is available online[4]. The problem generator creates random multi-sack to multi-stack problems. All blocks are specified in both the initial state and the goal state.

A `blocksworld` domain was used in IPC 2 but the problem set created for the competition is not used. The IPC 2 problems are flawed as each problem involves multi-stack to single-stack problems. A direct comparison is also not justified as I have used the 3 operator version of the domain whereas the 4 operator version was used for the competition.

The problems range from 6 to 9 blocks, with ten problems within each size. None

[4]Source is available at: http://users.rsise.anu.edu.au/∼jks/bwstates.html

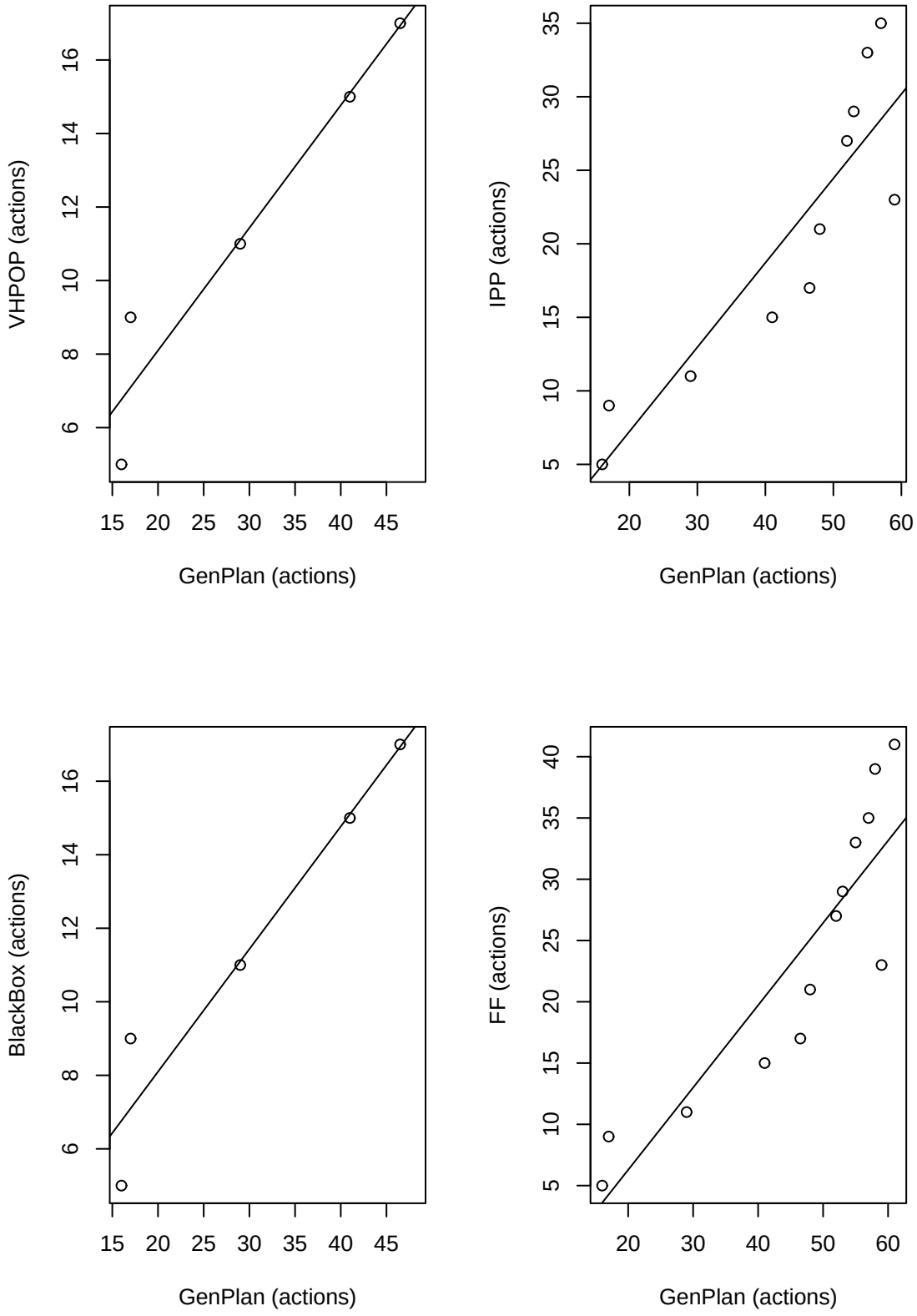



Figure 8.14: Regression plots for plan quality for `gripper`

of the problems from the preceding chapter were used and all the problems are new instances.

- let $B$ be the number of blocks
  - number of `move-b-to-b(?b1 ?b2 ?b3)` actions $= B*(B-1)*(B-2)$
  - number of `move-b-to-t(?b1 ?b2)` actions $= B*(B-1)$
  - number of `move-t-to-b(?b1 ?b2)` actions $= B*(B-1)$
- Action Space: 180-648 actions

#### 8.2.5.1 Results

Referring to Figure 8.15 it is difficult to draw any conclusions. In general, GENPLAN is taking the longest to solve problems in terms of raw efficiency. The test planners can be ordered from best to last as: FF, IPP, BB, VHPOP. The VHPOP curves are missing points as the planner fails to solve a number of problems even within the increased memory bound of 2GB.

The original GENPLAN results suggested that genetic planning was good at solving `blocksworld` problems [Westerberg and Levine, 2000]. The results here do not support this conclusion, in part this is due to the harder domain encoding (3 operators versus 2 operators with local variables), and due to using random problems. The issue of domain encoding, for `blocksworld`, is returned to in Chapter 9. Referring to Figure 8.15, the timing results vary because they are all instances of a particular class i.e. 7 block problems. Within that particular class there are easier and harder problems. As `blocksworld` has both variable sized benches and variable size local minima this make `blocksworld` an interesting domain. The effects of both search issues are examined in Chapter 9.

Plan quality results are given in Figure 8.16. GENPLAN is generally producing the lowest quality of plans for this domain, but occasionally VHPOP, BB, and IPP produce a plan of lower quality than that of GENPLAN.

#### 8.2.5.2 Evaluation - Efficiency

Referring to Figure 8.15, it is difficult to formulate any intuitions from the figure except that GENPLAN appears to be performing less well when compared to BB, IPP, and FF. For this analysis results from two problems, 9-001 and 9-008, are removed due to ceiling effects.

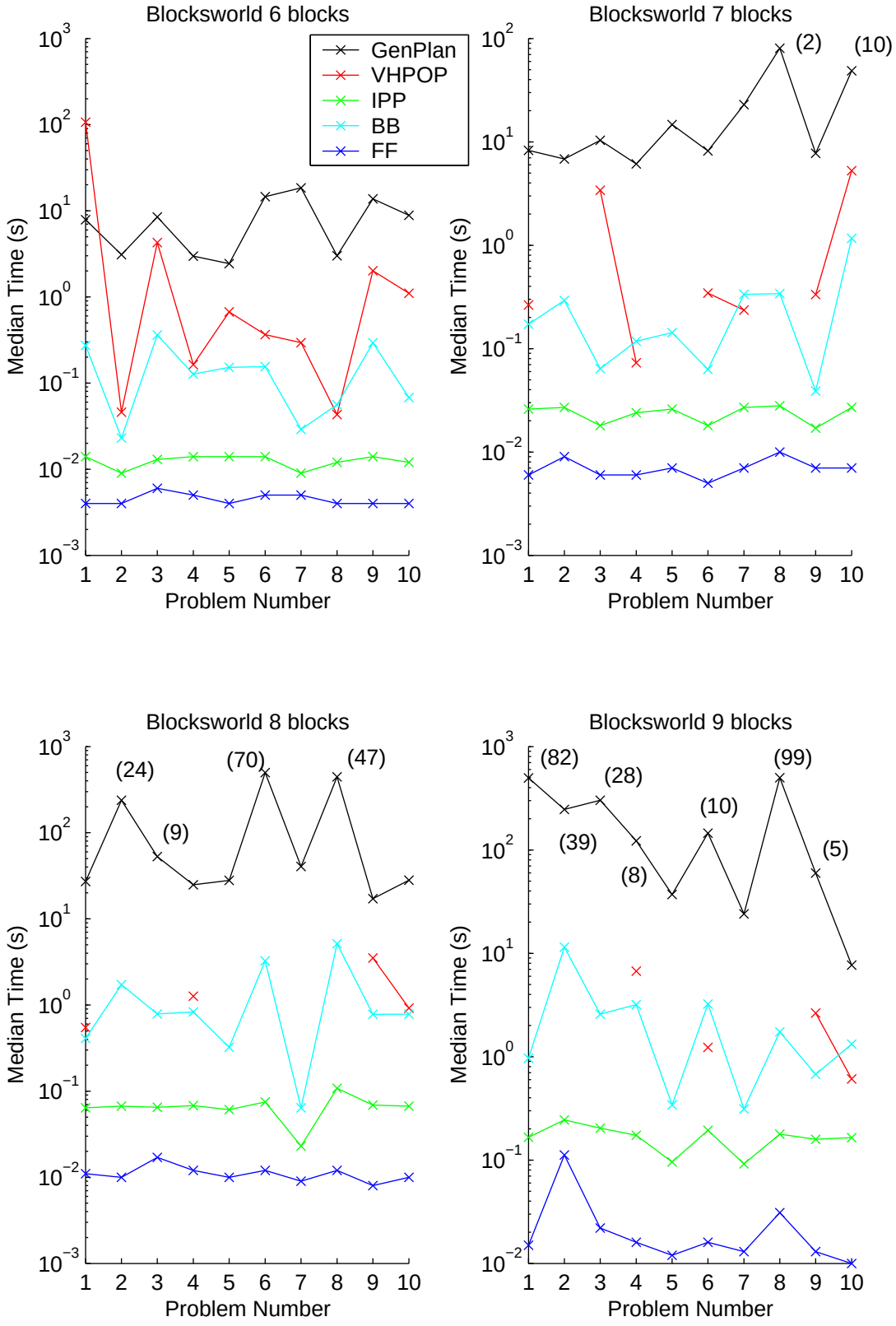



Figure 8.15: Efficiency results for `blocksworld-3ops`

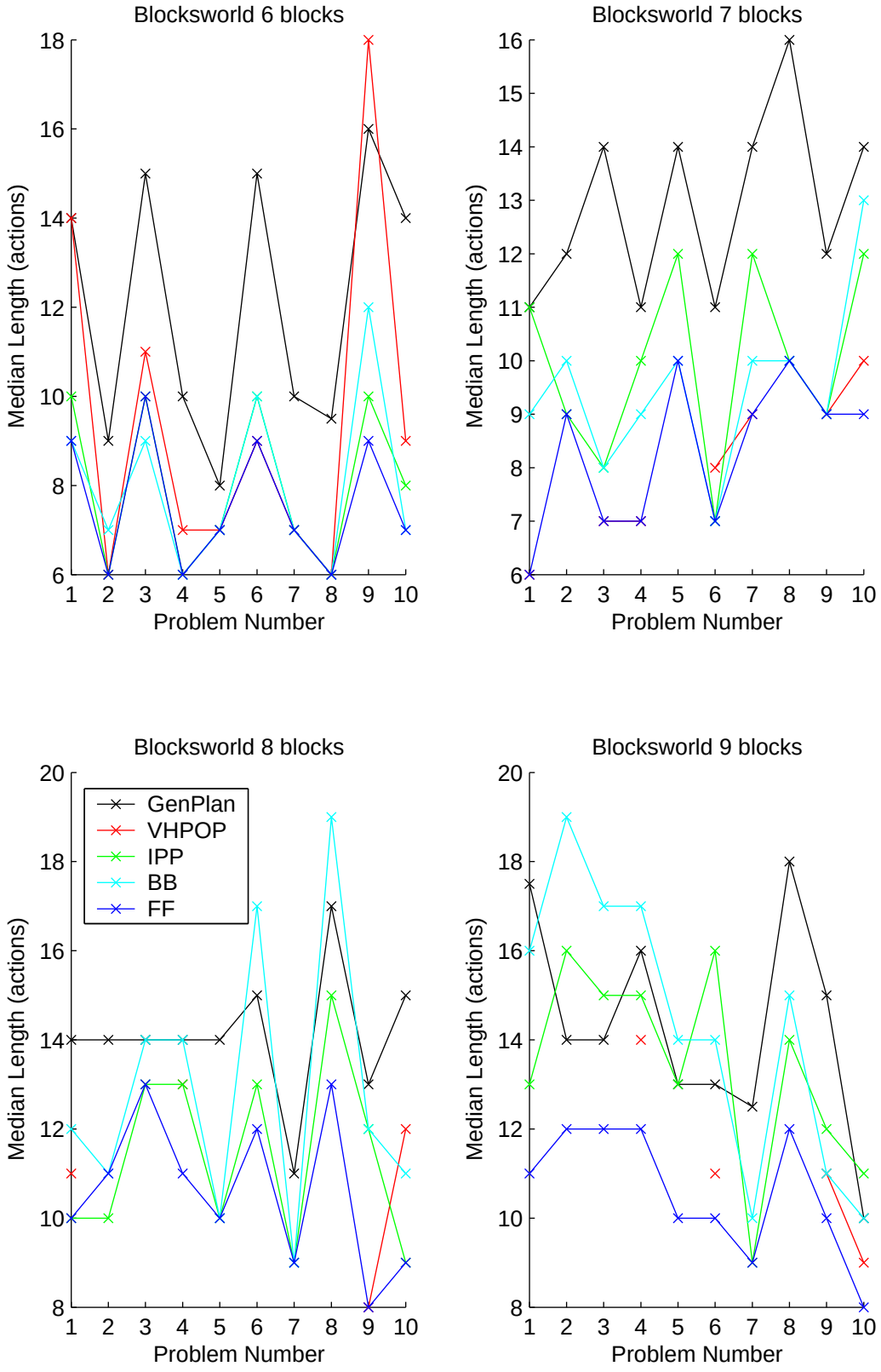



Figure 8.16: Plan quality results for `blocksworld-3ops`

- **Null Hypothesis** $H_{n0}$: The relationship of solution time between VHPOP and GENPLAN is linear.
  **Alternative Hypothesis**: The relationship of solution time between VHPOP and GENPLAN contains a significant degree of non-linearity.

- **Null Hypothesis** $H_{n1}$: The relationship of solution time between IPP and GENPLAN is linear.
  **Alternative Hypothesis**: The relationship of solution time between IPP and GENPLAN contains a significant degree of non-linearity.

- **Null Hypothesis** $H_{n2}$: The relationship of solution time between BB and GENPLAN is linear.
  **Alternative Hypothesis**: The relationship of solution time between BB and GENPLAN contains a significant degree of non-linearity.

- **Null Hypothesis** $H_{n3}$: The relationship of solution time between FF and GENPLAN is linear.
  **Alternative Hypothesis** The relationship of solution time between FF and GENPLAN contains a significant degree of non-linearity.

The discussion refers to both Table 8.13 and Figure 8.17.

- There is insufficient evidence to reject $H_{n0}$.

- There is sufficient evidence to reject $H_{n1}$. There is a significant degree of non-linearity and due to the shape of the curve GENPLAN is scaling worse than IPP.

- There is sufficient evidence to reject $H_{n2}$. There is a significant degree of non-linearity and due to the shape of the curve GENPLAN is scaling worse than BB.

- There is sufficient evidence to reject $H_{n3}$. There is a significant degree of non-linearity and due to the shape of the curve GENPLAN is scaling worse than FF.

The non-linearity test states that GENPLAN is scaling worse than BB, IPP, and FF. The test between GENPLAN and VHPOP is inconclusive but in defence of GENPLAN it was able to solve more problems than VHPOP.

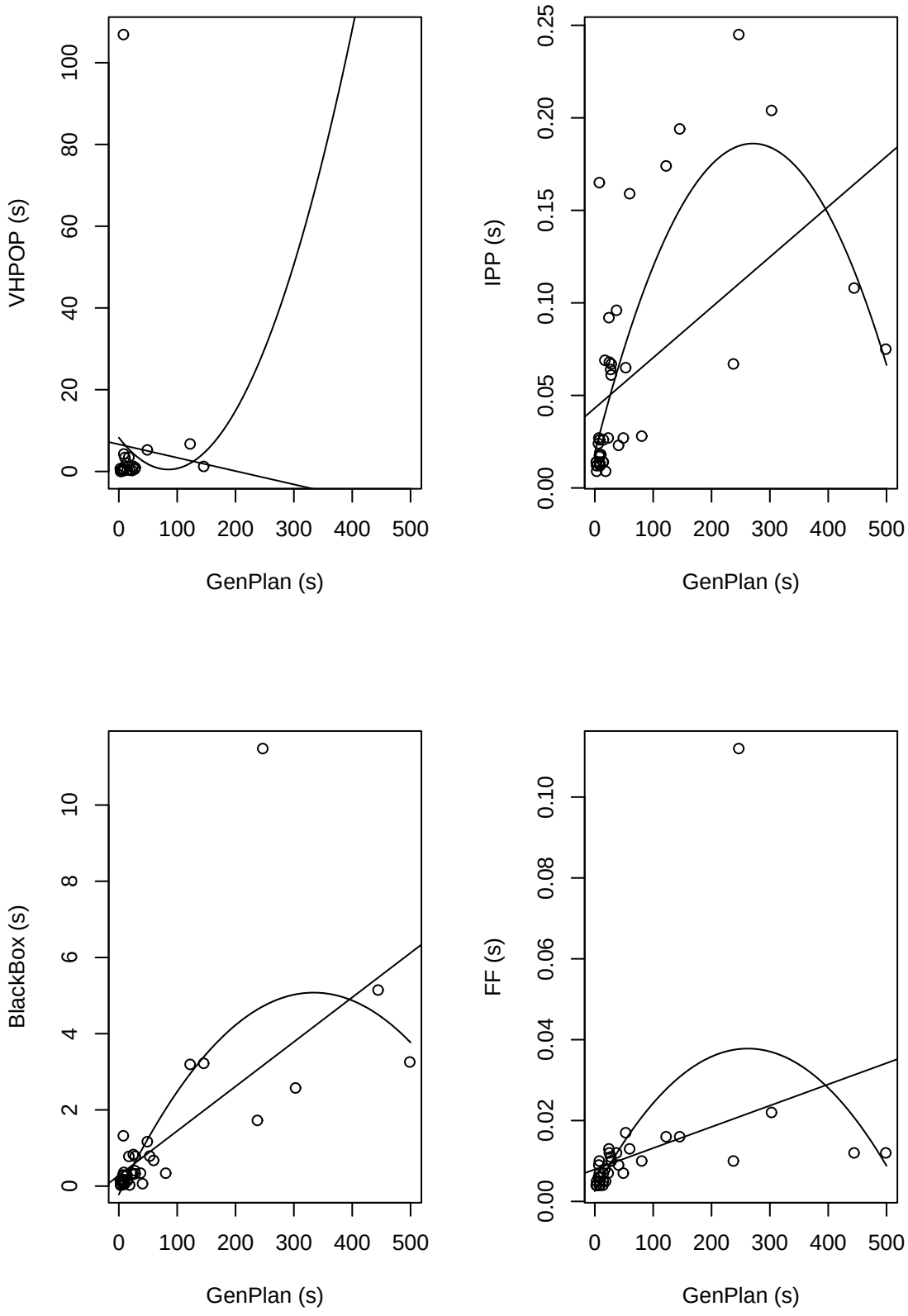



Figure 8.17: Efficiency Regression Plots for `blocksworld-3ops`

| Model - Efficiency | $R^2$ Linear | $R^2$ Non-linear | P-value |
|---|---|---|---|
| $VHPOP \sim GenPlan$ | 0.002919 | 0.006254 | 0.793 |
| $IPP \sim GenPlan$ | 0.2642 | 0.5684 | $<$0.0001 |
| $BB \sim GenPlan$ | 0.4502 | 0.5725 | 0.00321 |
| $FF \sim GenPlan$ | 0.1331 | 0.3415 | 0.0021 |

Table 8.13: Statistical data for Figure 8.17

#### 8.2.5.3 Evaluation - Quality

Referring to Figure 8.16 it is difficult to draw any intuitions about the scaling behaviour of plan quality for the planners. The results will be examined in conjunction with the non-linearity test. For the this analysis results from two problems, 9-001 and 9-008, are removed due to ceiling effects caused by GENPLAN.

- **Null Hypothesis** $H_{n0}$: The relationship of plan quality between VHPOP and GENPLAN is linear.
  **Alternative Hypothesis**: The relationship of plan quality between VHPOP and GENPLAN contains a significant degree of non-linearity.

- **Null Hypothesis** $H_{n1}$: The relationship of plan quality between IPP and GENPLAN is linear.
  **Alternative Hypothesis**: The relationship of plan quality between IPP and GENPLAN contains a significant degree of non-linearity.

- **Null Hypothesis** $H_{n2}$: The relationship of plan quality between BB and GENPLAN is linear.
  **Alternative Hypothesis**: The relationship of plan quality between BB and GENPLAN contains a significant degree of non-linearity.

- **Null Hypothesis** $H_{n3}$: The relationship of plan quality between FF and GENPLAN is linear.
  **Alternative Hypothesis** The relationship of plan quality between FF and GENPLAN contains a significant degree of non-linearity.

The discussion refers to both Table 8.14 and Figure 8.18.

| Model - Quality | $R^2$ Linear | $R^2$ Non-linear | P-value |
|---|---|---|---|
| $VHPOP \sim GenPlan$ | 0.5665 | 0.6448 | 0.0432 |
| $IPP \sim GenPlan$ | 0.3519 | 0.3681 | 0.350 |
| $BB \sim GenPlan$ | 0.3875 | 0.3887 | 0.792 |
| $FF \sim GenPlan$ | 0.5226 | 0.5226 | 0.984 |

Table 8.14: Statistical data for Figure 8.18

- There is sufficient evidence to reject $H_0$. There is a significant degree of non-linearity and due to the shape of the curve GENPLAN is scaling better than VHPOP.
- There is insufficient evidence to reject $H_{n1}$.
- There is insufficient evidence to reject $H_{n2}$.
- There is insufficient evidence to reject $H_{n3}$.

A surprising result of this test is that there is a significant non-linearity between GENPLAN and VHPOP and in favour of GENPLAN. There was no significant difference found between comparing the quality results of GENPLAN with IPP, BB, and FF.

### 8.2.6 Numeric Mystery

The numeric `mystery` domain is a transportation domain with the additional complication of fuel and carrying restrictions. The problems for this domain can be varied in a number of ways. The three main variables are vehicles, locations, and cargos. Like the `briefcase` domain, I fixed the number of carriers (vehicles) to 1. It is important to note that locations are not guaranteed to be fully connected to each other, but there is a guaranteed path between all locations. The problems were generated using my own random problem generator. The changes made to the domain file mean that it is not justified to include the IPC 1 problems, see Chapter 6 for the changes made. Cargos and the vehicle can be assigned to any location in the initial state. For the goal state cargos can not be assigned the same location as their initial state and no goal location is assigned to the vehicle meaning there are no local minima.

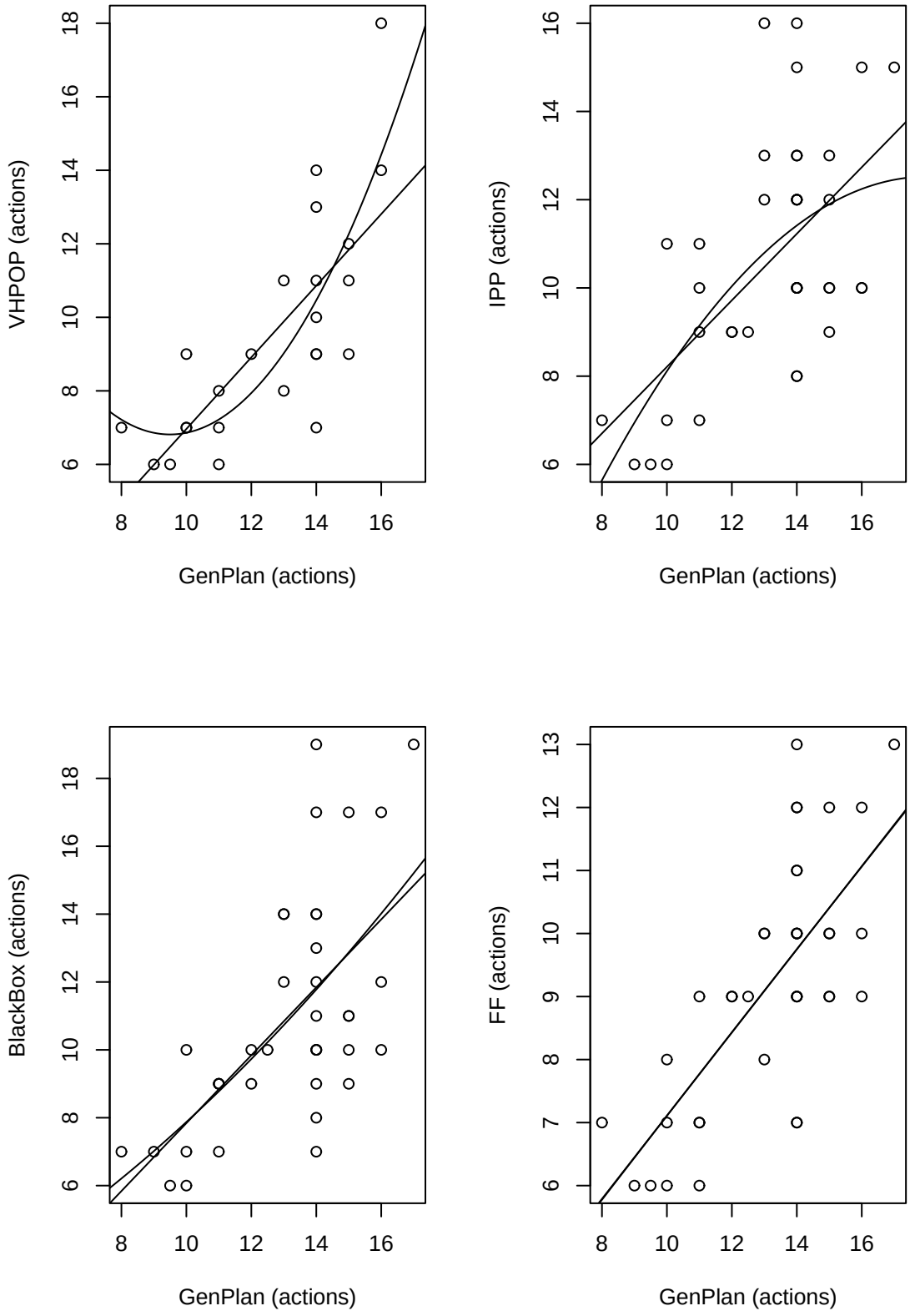



Figure 8.18: Regression plots for plan quality for `blocksworld-3ops`

The number of locations were varied from 5 to 8 and for each number of locations the number of cargos was varied from 1 up to a maximum of 8. Another important consideration is where to fix the resource requirements. I choose the resource requirements to be quite lenient for these problems. Each location gets the amount of fuel equivalent to the number of locations in the domain and the vehicle gets a cargo capacity equal to the number of cargos. The effect of limited cargo capacity has already been examined in the `gripper` section.

- Let $V$ be the number of vehicles, let $L$ be the number of locations, let $C$ be the number of cargos.
  - number of `move(?v ?l1 ?l2)` actions $= V * L * (L-1)$
  - number of `load(?c ?v ?l)` actions $= V * C * L$
  - number of `unload(?c ?v ?l)` actions $= V * C * L$
- Action Space: 26-104 actions

#### 8.2.6.1 Results

Referring to Figure 8.19, GENPLAN presents poor scaling for this domain whereas FF presents good scaling for all the problems sets. As this domain is now a numeric domain, FF is the only planner from the four test planners that supports this PDDL requirement.

GENPLAN's somewhat variable efficiency results can be explained by the fact that the locations are not fully connected. This in turn makes variable sized benches within the search space, unlike the constant minimal size of 3 for `briefcase`. Limited fuel may also be an issue for the larger problems. When the fuel runs out at locations this can create dead-ends in the search space. The effect of this resource requirement is highlighted in Chapter 9.

Referring to Figure 8.20, GENPLAN is producing plans a constant size longer than those of FF and GENPLAN is occasionally producing plans of higher quality.

#### 8.2.6.2 Evaluation - Efficiency

The scaling behaviour of GENPLAN appears worse than that of FF. For this analysis results from two problems, 8 locations-3 cargos and 8 locations-5 cargos, are removed due to ceiling effects caused by GENPLAN.

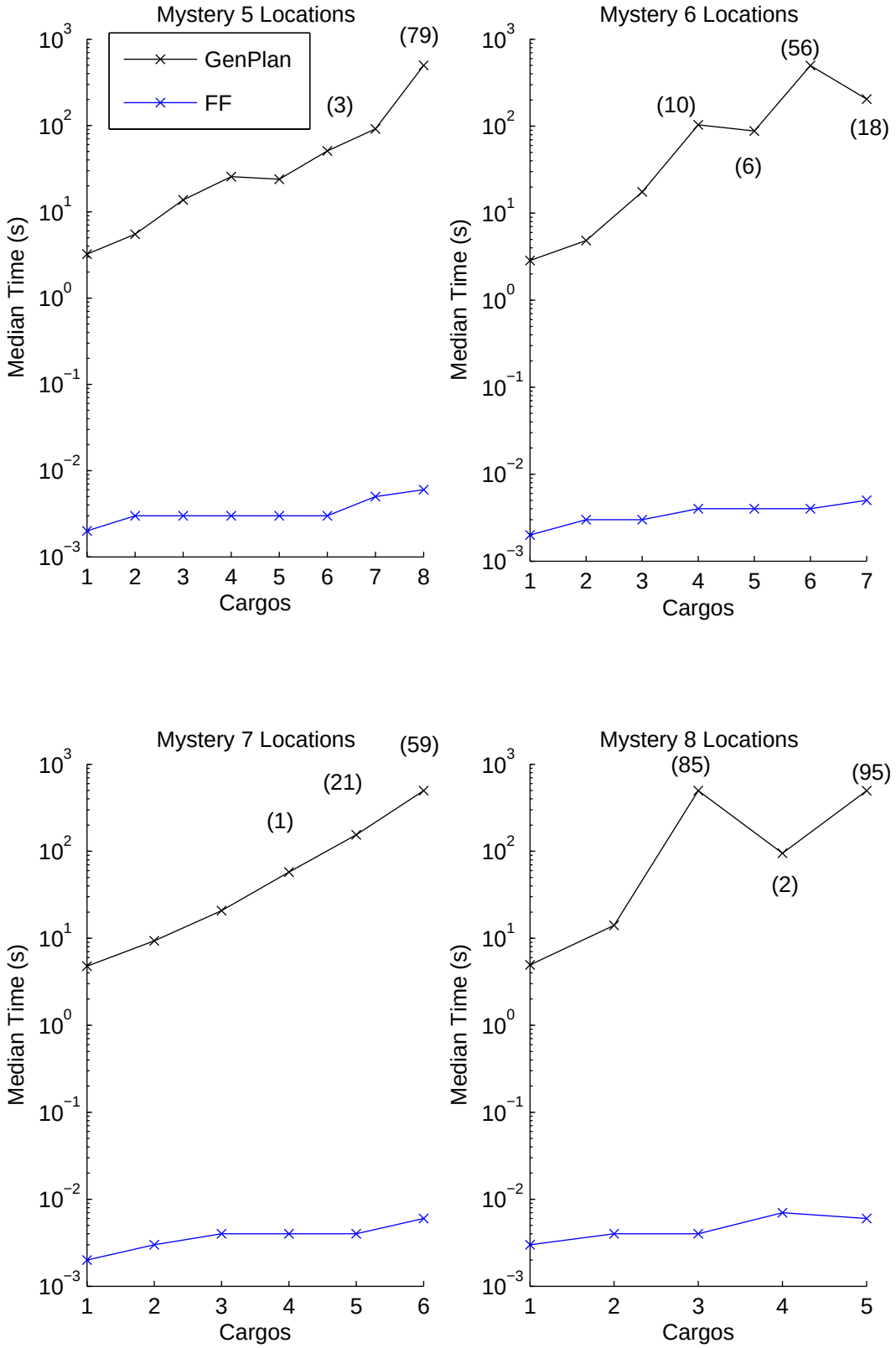



Figure 8.19: Efficiency results for `mystery`

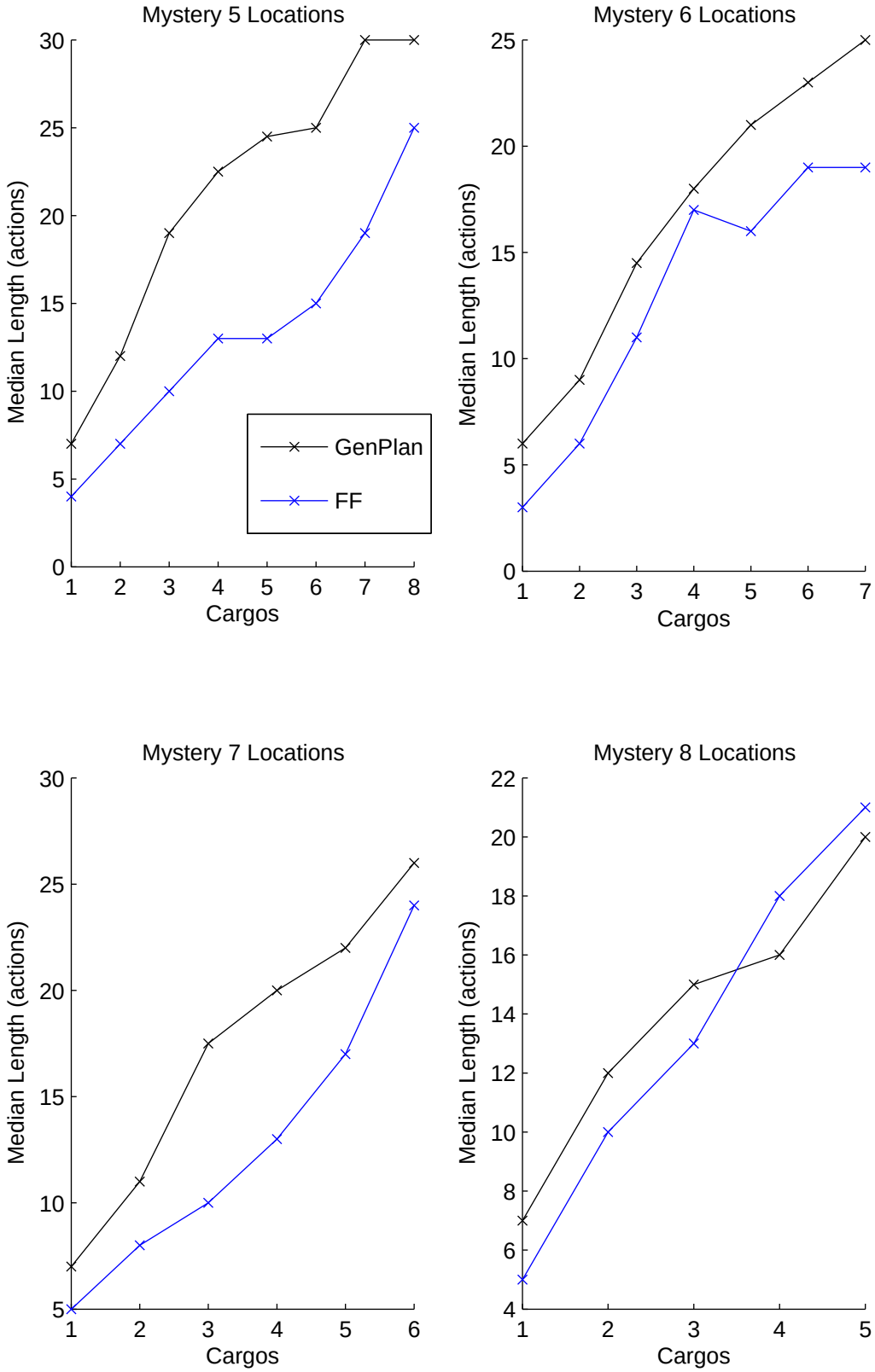



Figure 8.20: Plan quality results for `mystery`

| Model - Efficiency | $R^2$ Linear | $R^2$ Non-linear | P-value |
|---|---|---|---|
| $FF \sim GenPlan$ | 0.4441 | 0.5821 | 0.02155 |

Table 8.15: Statistical data for Figure 8.21

| Model - Quality | $R^2$ Linear |
|---|---|
| $FF \sim GenPlan$ | 0.7938 |

Table 8.16: Statistical data for Figure 8.22

- **Null Hypothesis** $H_{n0}$: The relationship of solution time between FF and GENPLAN is linear.
  **Alternative Hypothesis** The relationship of solution time between FF and GENPLAN contains a significant degree of non-linearity.

The discussion refers to both Table 8.15 and Figure 8.21.

- There is sufficient evidence to reject $H_{n0}$. There is a significant degree of non-linearity and due to the shape of the curve GENPLAN is scaling worse than FF.

The non-linearity test has confirmed that GENPLAN is scaling worse than FF.

#### 8.2.6.3 Evaluation - Quality

Referring to Table 8.16 and Figure 8.22 the apparent equivalent scaling of plan quality of the two planners is confirmed by the figure and by the good $R^2$ value. For this analysis the results from two problems, 8 locations-3 cargos and 8 locations-5 cargos, are again not included. In addition, the non-linearity test reported that there is insufficient evidence to claim a significant non-linearity in the data comparing GENPLAN and FF.

### 8.2.7 Miconic-STRIPS

This domain involves moving passengers between floors using an elevator. There are two variables to vary for this domain: the number of floors and the number of passengers. The problems were created using the random problem generator from Joerg

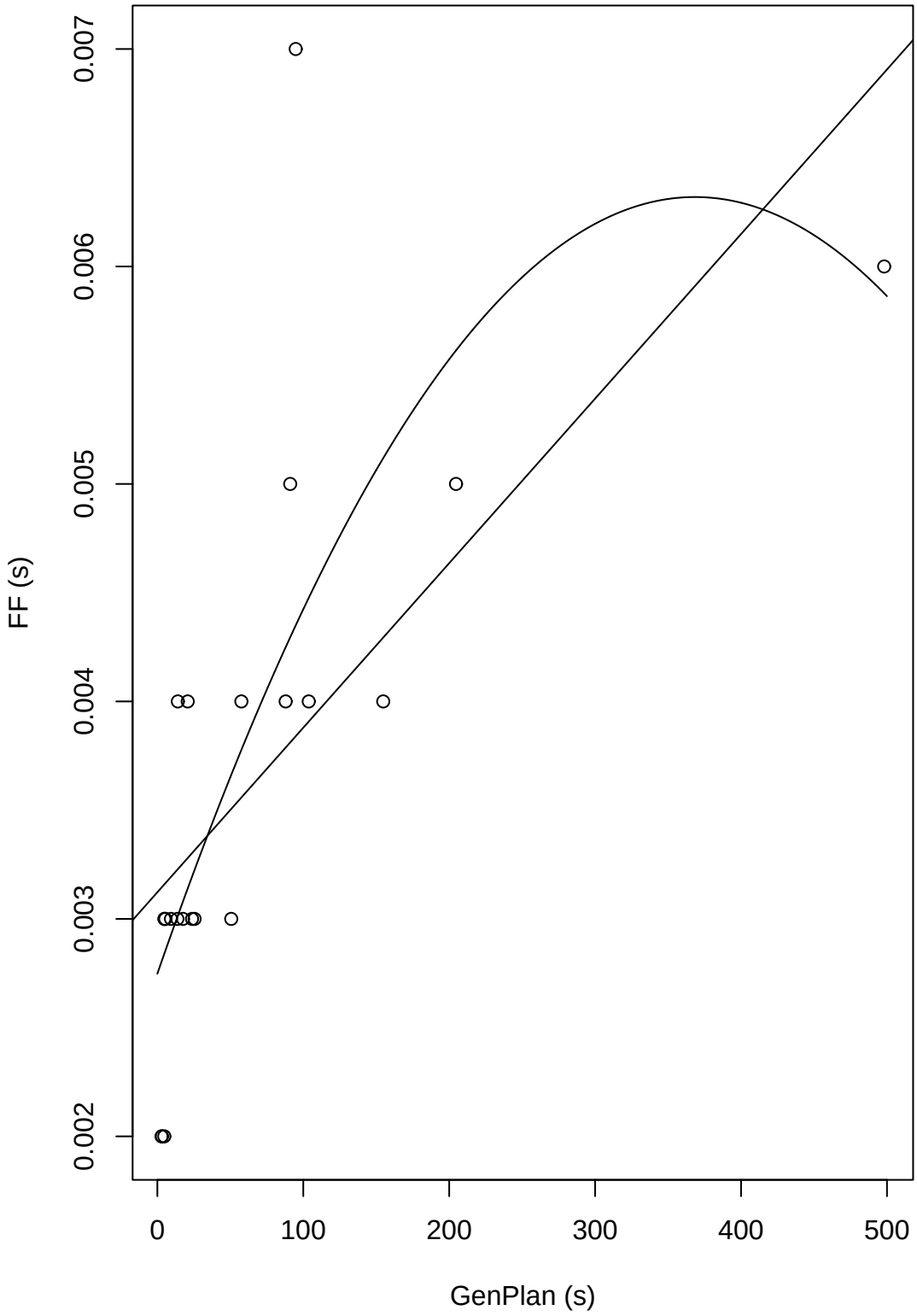



Figure 8.21: Efficiency Regression Plots for Numeric `mystery`

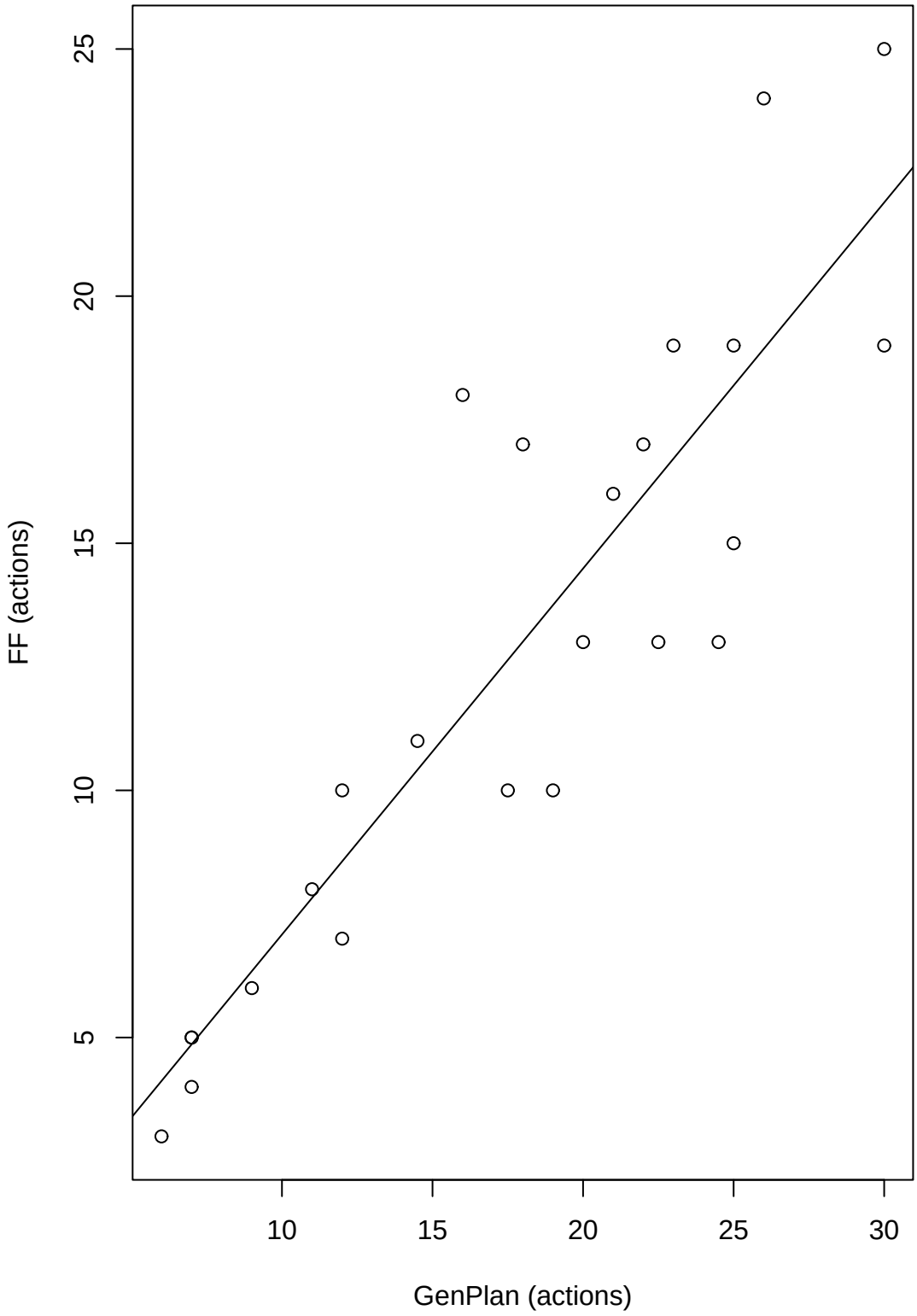



Figure 8.22: Regression plots for plan quality for Numeric `mystery`

Hoffman's domain repository[5]. These problems are created such that no passenger has already been served and there is no goal location for the elevator so that the local minima value for this domain is 0. A goal location is specified for every passenger. Results are also generated from a selection of problems used in IPC 2. The naming convention used for the IPC problems is to first give the original name and the numbers in parenthesis indicate the number of floors and passengers respectively.

The random problems were divided into four sets: 6, 7, 8, and 9 floors respectively. For each number of floors the number of passengers was varied from 1 to 7.

- Let $F$ be the number of floors, and let $P$ be the number of passengers
  - number of `board(?f ?p)` actions $= F * B$
  - number of `depart(?f ?p)` actions $= F * B$
  - number of `up(?f1 ?f2)` actions $= F * (F-1)$
  - number of `down(?f1 ?f2)` actions $= F * (F-1)$
- Action Space: 72-270 actions

#### 8.2.7.1 Results

Referring to Figure 8.23, GENPLAN is scaling poorly for this domain. This plot also clearly demonstrates the effect that large numbers of failures can have on the median time with GENPLAN's curves flattening slightly for the 8 and 9 floor results. The scaling of BB and IPP appears to be the same or worse than that of GENPLAN's and FF and VHPOP appear to be scaling better. The results in Table 8.17, which make use of the IPC instances, also support the earlier intuitions.

The main difficulty in this domain are the larger action spaces due to the fact there are 4 operators in the domain. Due to the way in which random actions are made without reference to the problem file means that the reported action space values include many redundant actions. The true range should be between 44-104 actions. This domain highlights the need for an improved domain pre-processing algorithm for GENPLAN. This extra action space is hampering search when the action space range for `briefcase` is considered. This action space issue is returned to in Chapter 9. The domain contains no local minima, dead-ends, and one infinite carrying resource.

Referring to Figure 8.24, GENPLAN is producing plans a constant size longer than those plans produced by the other planners. FF, IPP, and BB are producing the highest

[5] http://www.mpi-sb.mpg.de/~hoffmann/ff-domains.html

quality plans with VHPOP producing slightly longer plans than the three other test planners. The results in Table 8.18, which make use of the IPC instances, also support the earlier discussion. The only exception being the 2 floor, 1 passenger problems where GENPLAN produces overly long plans.

#### 8.2.7.2 Evaluation - Efficiency

The discussion in the previous section speculated that GENPLAN is scaling about the same as BB and IPP and scaling worse than VHPOP and BB. These intuitions will be explored using the 6 floor problems to avoid the ceiling effects in the GENPLAN results.

- **Null Hypothesis** $H_{n0}$: The relationship of solution time between VHPOP and GENPLAN is linear for the 6 floor problems.
  **Alternative Hypothesis**: The relationship of solution time between VHPOP and GENPLAN contains a significant degree of non-linearity for the 6 floor problems.

- **Null Hypothesis** $H_{n1}$: The relationship of solution time between IPP and GENPLAN is linear for the 6 floor problems.
  **Alternative Hypothesis**: The relationship of solution time between IPP and GENPLAN contains a significant degree of non-linearity for the 6 floor problems.

- **Null Hypothesis** $H_{n2}$: The relationship of solution time between BB and GENPLAN is linear for the 6 floor problems.
  **Alternative Hypothesis**: The relationship of solution time between BB and GENPLAN contains a significant degree of non-linearity for the 6 floor problems.

- **Null Hypothesis** $H_{n3}$: The relationship of solution time between FF and GENPLAN is linear for the 6 floor problems.
  **Alternative Hypothesis** The relationship of solution time between FF and GENPLAN contains a significant degree of non-linearity for the 6 floor problems.

The discussion refers to both Table 8.19 and Figure 8.25.

- There is sufficient evidence to reject $H_{n0}$. There is a significant degree of non-linearity and due to the shape of the curve GENPLAN is scaling worse than VHPOP for the 6 floor problems.

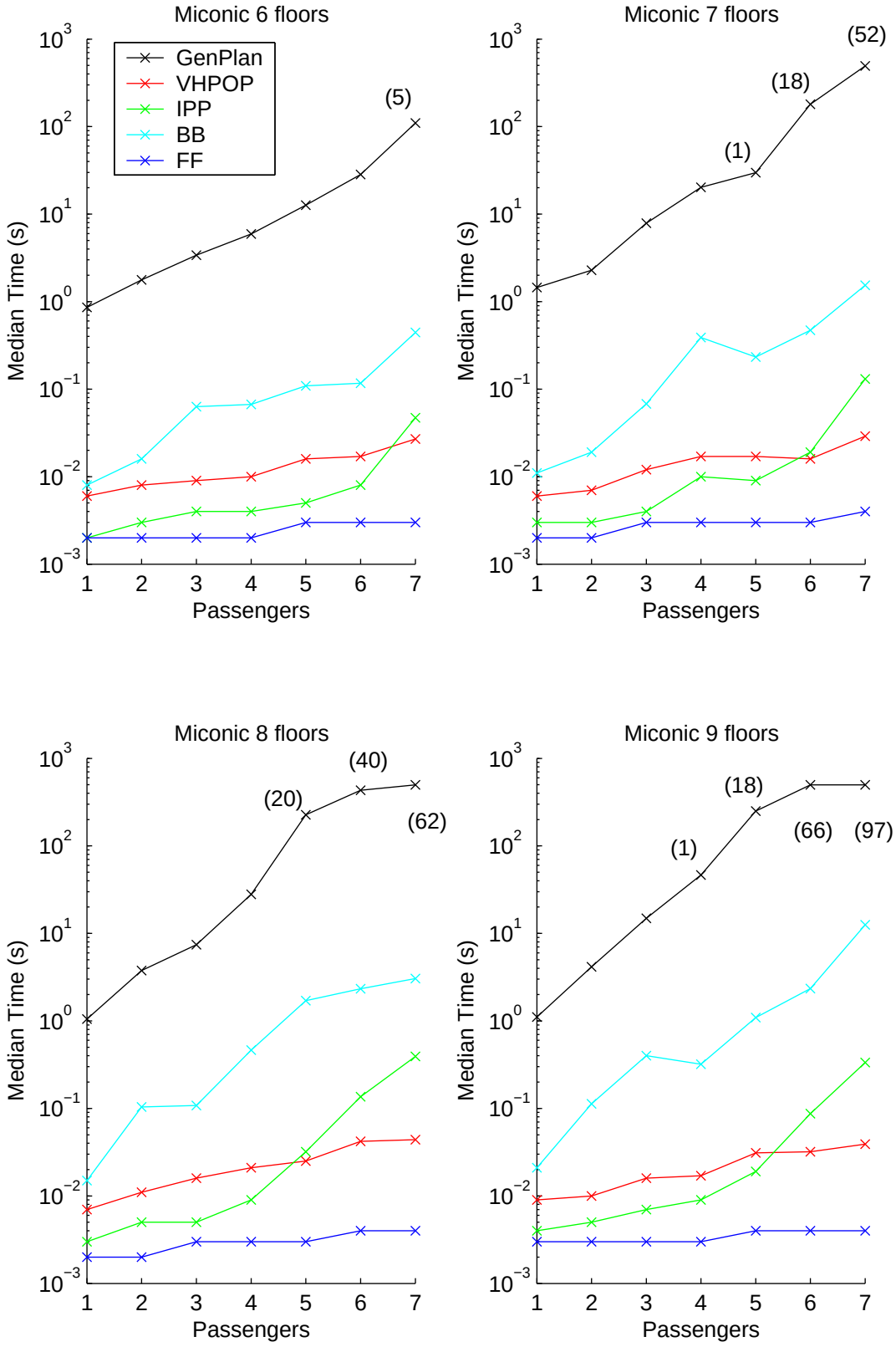



Figure 8.23: Efficiency results for `miconic-STRIPS`

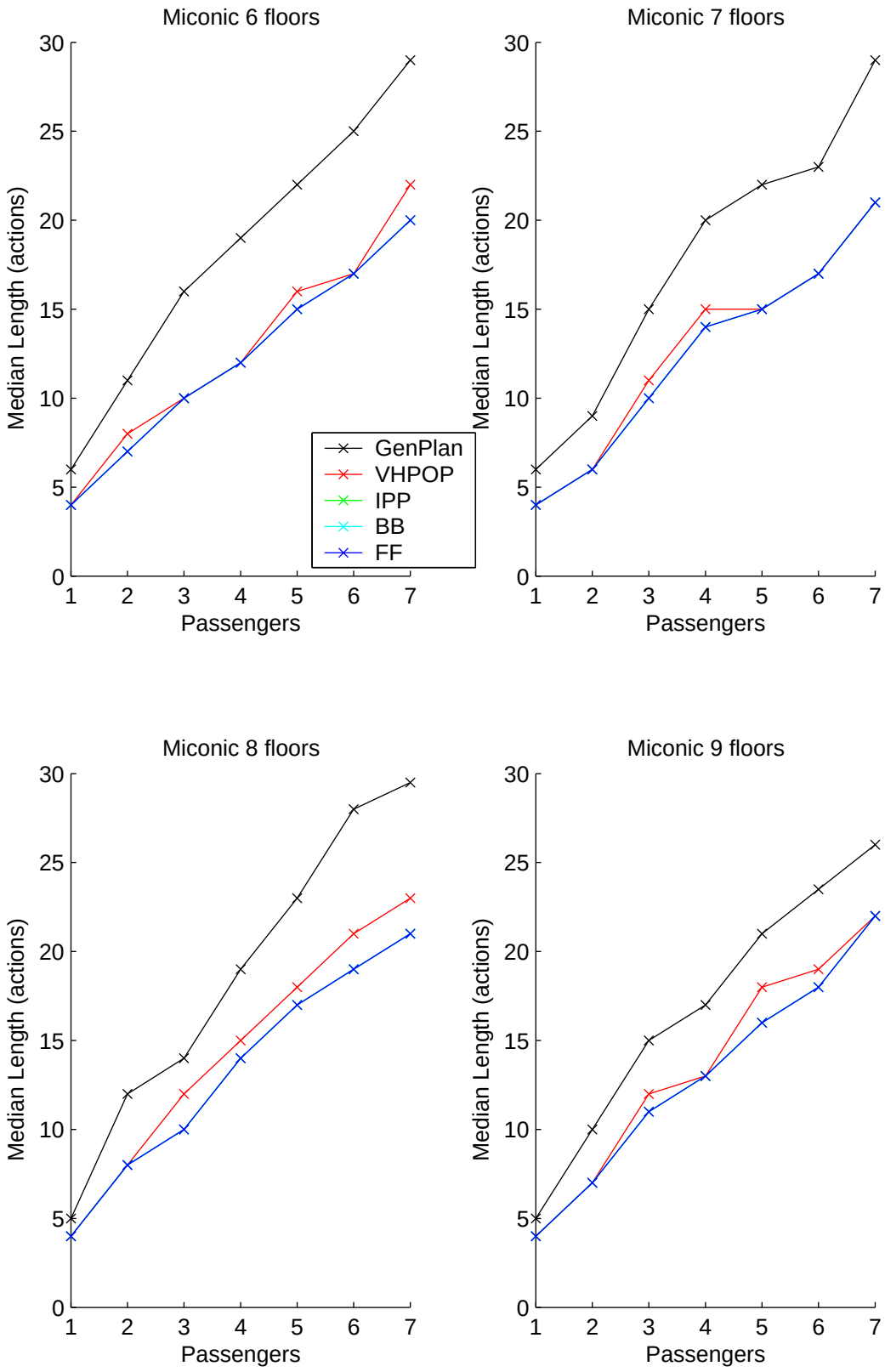



Figure 8.24: Plan quality results for `miconic-STRIPS`

| Problem | GenPlan | VHPOP | IPP | BB | FF |
|---|---|---|---|---|---|
| s1-0(2-1) | 0.96 | 0.004 | 0.002 | 0.002 | 0.003 |
| s1-1(2-1) | 0.96 | 0.004 | 0.002 | 0.002 | 0.002 |
| s1-2(2-1) | 0.96 | 0.004 | 0.001 | 0.003 | 0.002 |
| s1-3(2-1) | 0.96 | 0.004 | 0.002 | 0.002 | 0.002 |
| s1-4(2-1) | 0.96 | 0.004 | 0.002 | 0.003 | 0.002 |
| s2-0(4-2) | 1 | 0.006 | 0.002 | 0.007 | 0.003 |
| s2-1(4-2) | 0.99 | 0.006 | 0.002 | 0.010 | 0.003 |
| s2-2(4-2) | 1.19 | 0.006 | 0.002 | 0.010 | 0.002 |
| s2-3(4-2) | 1.01 | 0.006 | 0.002 | 0.009 | 0.003 |
| s2-4(4-2) | 1.28 | 0.006 | 0.002 | 0.007 | 0.002 |
| s3-0(6-3) | 3.55 | 0.013 | 0.003 | 0.048 | 0.003 |
| s3-1(6-3) | 3.57 | 0.010 | 0.004 | 0.087 | 0.003 |
| s3-2(6-3) | 3.47 | 0.026 | 0.004 | 0.053 | 0.003 |
| s3-3(6-3) | 3.53 | 0.010 | 0.004 | 0.072 | 0.003 |
| s3-4(6-3) | 3.5 | 0.014 | 0.004 | 0.055 | 0.003 |
| s4-0(8-4) | 36.54 | 0.020 | 0.009 | 0.484 | 0.004 |
| s4-1(8-4) | 17.16 | 0.021 | 0.009 | 0.379 | 0.004 |
| s4-2(8-4) | 33.19 | 0.022 | 0.014 | 1.668 | 0.004 |
| s4-3(8-4) | 63.17 (4) | 0.022 | 0.014 | 1.543 | 0.004 |
| s4-4(8-4) | 63.19 | 0.021 | 0.015 | 1.714 | 0.004 |
| s5-0(10-5) | 497.56 (53) | 0.032 | 0.049 | 3.797 | 0.005 |
| s5-1(10-5) | 498.33 (70) | 0.036 | 0.077 | 6.145 | 0.005 |
| s5-2(10-5) | 288.64 (23) | 0.036 | 0.014 | 0.614 | 0.005 |
| s5-3(10-5) | 498.24 (69) | 0.029 | 0.057 | 3.332 | 0.005 |
| s5-4(10-5) | 497.94 (59) | 0.045 | 0.056 | 8.175 | 0.005 |

Table 8.17: Efficiency results on IPC instances

| Problem | GenPlan | VHPOP | IPP | BB | FF |
|---|---|---|---|---|---|
| s1-0(2-1) | 13 | 4 | 4 | 4 | 4 |
| s1-1(2-1) | 13 | 3 | 3 | 3 | 3 |
| s1-2(2-1) | 12 | 4 | 4 | 4 | 4 |
| s1-3(2-1) | 14 | 4 | 4 | 4 | 4 |
| s1-4(2-1) | 13 | 4 | 4 | 4 | 4 |
| s2-0(4-2) | 10 | 8 | 7 | 7 | 7 |
| s2-1(4-2) | 10.5 | 7 | 7 | 7 | 7 |
| s2-2(4-2) | 10 | 7 | 7 | 7 | 7 |
| s2-3(4-2) | 10 | 8 | 7 | 7 | 7 |
| s2-4(4-2) | 10 | 7 | 7 | 7 | 7 |
| s3-0(6-3) | 17 | 12 | 10 | 10 | 10 |
| s3-1(6-3) | 17 | 11 | 11 | 11 | 11 |
| s3-2(6-3) | 16 | 11 | 10 | 10 | 10 |
| s3-3(6-3) | 16 | 10 | 10 | 10 | 10 |
| s3-4(6-3) | 16 | 11 | 10 | 10 | 10 |
| s4-0(8-4) | 19 | 15 | 14 | 14 | 14 |
| s4-1(8-4) | 17 | 13 | 13 | 13 | 13 |
| s4-2(8-4) | 19 | 15 | 15 | 15 | 15 |
| s4-3(8-4) | 20 | 15 | 15 | 15 | 15 |
| s4-4(8-4) | 20 | 15 | 15 | 15 | 15 |
| s5-0(10-5) | 22 | 17 | 17 | 17 | 17 |
| s5-1(10-5) | 22 | 17 | 17 | 17 | 17 |
| s5-2(10-5) | 22 | 17 | 15 | 15 | 15 |
| s5-3(10-5) | 22 | 17 | 17 | 17 | 17 |
| s5-4(10-5) | 22 | 19 | 18 | 18 | 18 |

Table 8.18: Plan Quality results on IPC instances

| Model - Efficiency | $R^2$ Linear | $R^2$ Non-linear | P-value |
|---|---|---|---|
| $VHPOP \sim GenPlan$ | 0.8586 | 0.9527 | 0.04769 |
| $IPP \sim GenPlan$ | 0.9821 | 0.9989 | 0.00135 |
| $BB \sim GenPlan$ | 0.9773 | 0.9774 | 0.9637 |
| $FF \sim GenPlan$ | 0.4123 | 0.8319 | 0.0341 |

Table 8.19: Statistical data for Figure 8.25

- There is sufficient evidence to reject $H_{n1}$. There is a significant degree of non-linearity and due to the shape of the curve GENPLAN is scaling better than IPP for the 6 floor problems.
- There is insufficient evidence to reject $H_{n2}$.
- There is sufficient evidence to reject $H_{n3}$. There is a significant degree of non-linearity and due to the shape of the curve GENPLAN is scaling worse than FF for the 6 floor problems.

The results from the non-linearity test nearly confirm all the earlier intuitions. GENPLAN is again scaling better than IPP for a limited range of problems. GENPLAN is again scaling worse than both VHPOP and FF for this domain. The comparison between BB and GENPLAN is inconclusive.

#### 8.2.7.3 Evaluation - Quality

The results in Figure 8.26 includes all the data present in Figure 8.24 except the results from problem 9-7 and all the results in Table 8.18. Referring to Figure 8.26 and Table 8.20, they both confirm the linear relationship of plan quality between GENPLAN and the four test planners due to the high $R^2$ values and from the figure itself. In addition, the non-linearity test reported that there is insufficient evidence to claim a significant non-linearity in the data for each comparison between GENPLAN and another test planner.

### 8.2.8 Satellite

The goal of this domain is take images of objects in space using a satellite equipped with the right lens. There are a variety of variables to vary for this domain. These

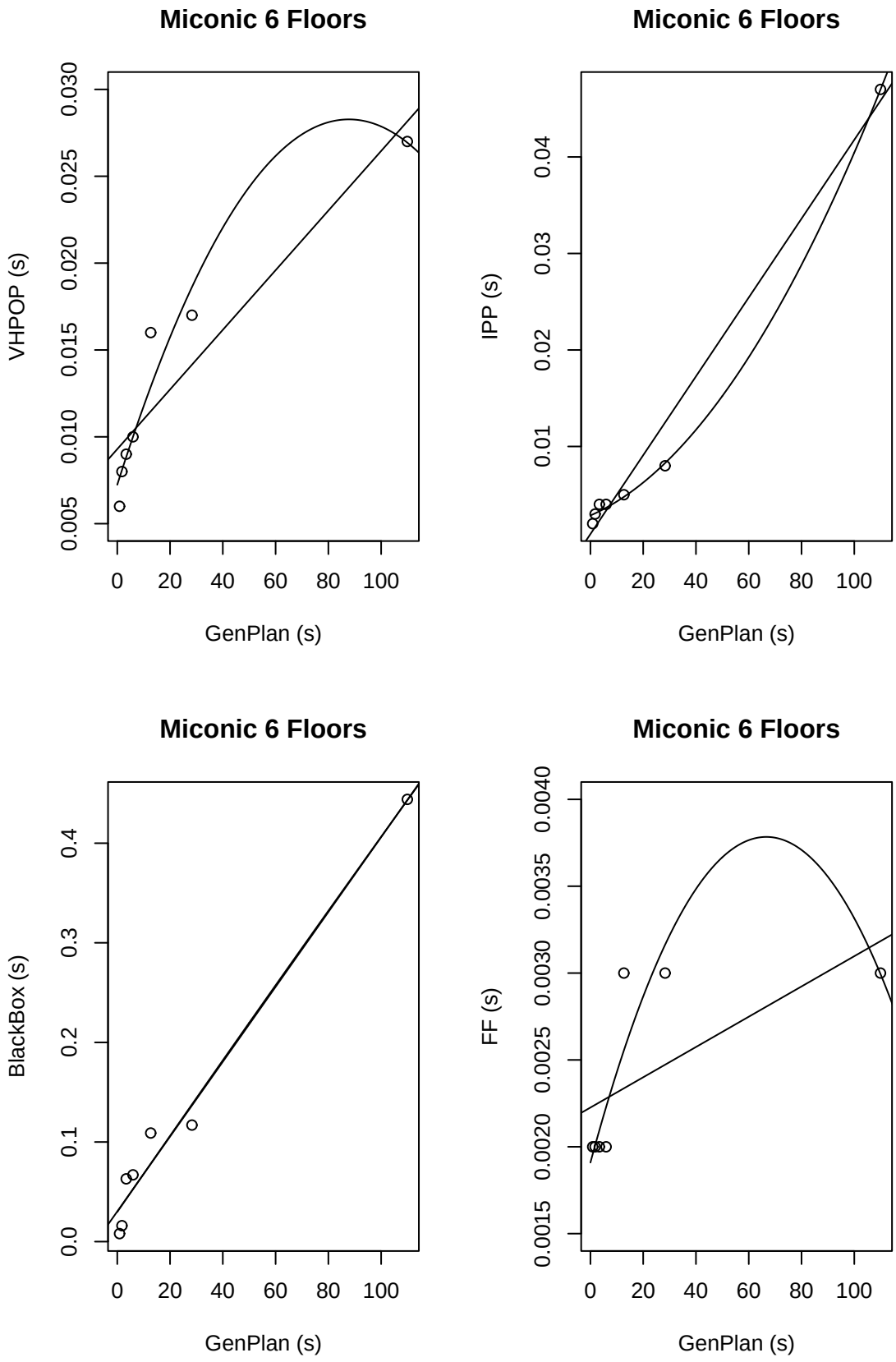



Figure 8.25: Efficiency Regression Plots for `miconic`

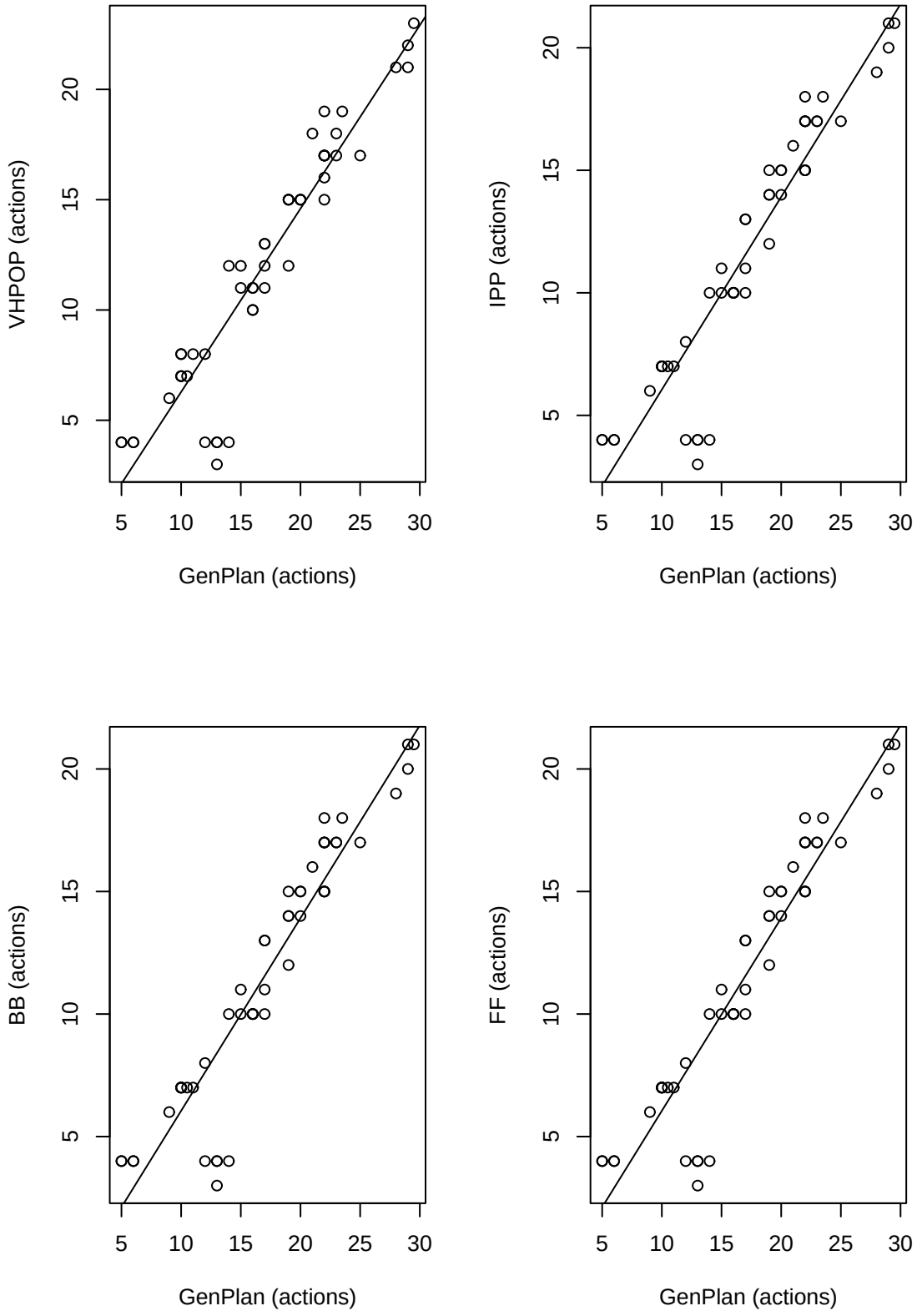



Figure 8.26: Regression plots for plan quality for `miconic`

| Model - Quality | $R^2$ Linear |
|---|---|
| $VHPOP \sim GenPlan$ | 0.8790 |
| $IPP \sim GenPlan$ | 0.8837 |
| $BB \sim GenPlan$ | 0.8837 |
| $FF \sim GenPlan$ | 0.8837 |

Table 8.20: Statistical data for Figure 8.26

include the number of satellites, the number of directions (objects), the number of instruments, and the number of modes (kinds of lenses). I used the random problem generator available from IPC 3.

The naming convention for these problems is: number of satellites, number of directions, maximum number of instruments per satellite, modes, and number of goals. None of the problems of the previous chapter were used in this chapter. I varied the number of images from 1-5. This in turn varies the total number of directions from 2-7. The number of modes was varied from 2 to 3. The number of satellites was varied from 1 to 2. This produced 4 classes of problems: 1-(2-7)-1-(2-3)-(1-5), and 2-(2-7)-1-(2-3)-(1-5). Goal directions where also given for the satellites creating 1 one-action local minima for the problems involving 1 satellite and 2 one-action local minima for the 2 satellite problems. Some problems from IPC 3 are also included but not all as they are very difficult.

- Let $S$ equal the number of satellites, $D$ the number of directions, $I$ the number of instruments, and $M$ the number of modes.
  - number of `turn_to(?s ?d ?d)` actions $= S*D*(D-1)$
  - number of `switch_on(?i ?s)` actions $= I*S$
  - number of `switch_off(?i ?s)` actions $= I*S$
  - number of `calibrate(?s ?i ?d)` actions $= S*I*D$
  - number of `take_image(?s ?d ?i ?m)` actions $= S*D*I*M$
- Range of actions: 17 - 204

| Problem | GenPlan | VHPOP | IPP | BB | FF |
|---|---|---|---|---|---|
| pfile1(1-7-3-3-3) | 28.63 | 0.011 | 0.004 | 0.006 | 0.003 |
| pfile2(1-8-3-3-5) | 189.67 (40) | 0.022 | 0.025 | 0.011 | 0.004 |
| pfile3(2-8-3-3-4) | 135.21 | 0.021 | 0.026 | 0.018 | 0.005 |
| pfile4(2-10-3-3-7) | 499.74 (100) | 0.078 | 5.709 | 0.024 | 0.007 |

Table 8.21: Efficiency results for `satellite` on selected IPC 3 instances

| Problem | GenPlan | VHPOP | IPP | BB | FF |
|---|---|---|---|---|---|
| pfile1(1-7-3-3-3) | 39 | 9 | 9 | 9 | 9 |
| pfile2(1-8-3-3-5) | 25 | 13 | 13 | 13 | 13 |
| pfile3(2-8-3-3-4) | 20 | 11 | 13 | 12 | 11 |
| pfile4(2-10-3-3-7) | - | 22 | 20 | 21 | 18 |

Table 8.22: Plan quality for `satellite` on selected IPC 3 instances

#### 8.2.8.1 Results

Referring to Figure 8.27, GENPLAN appears to be scaling worse than 3 of the other test planners: VHPOP, BB, and FF. However IPP is either scaling the same or worse than GENPLAN. FF presents the best raw efficiency and scaling behaviour. A handful of results are also presented in Table 8.21 using the IPC 3 instances.

The `satellite` domain makes use of multiple resources and the effect of having more than one satellite is to dramatically increase the size of the search space. The domain is also difficult because of its larger bench size and the inclusion of local minima in the problems.

Referring to Figure 8.28, GENPLAN is producing much longer plans than the four test planners. The four test planners produce a mixture of plan quality results with FF presenting the best overall results.

#### 8.2.8.2 Evaluation - Efficiency

The previous section speculated that GENPLAN may be scaling better than IPP. This will be investigated using specific problem sets from the random instances. It was also speculated that GENPLAN is scaling worse than the three other test planners and

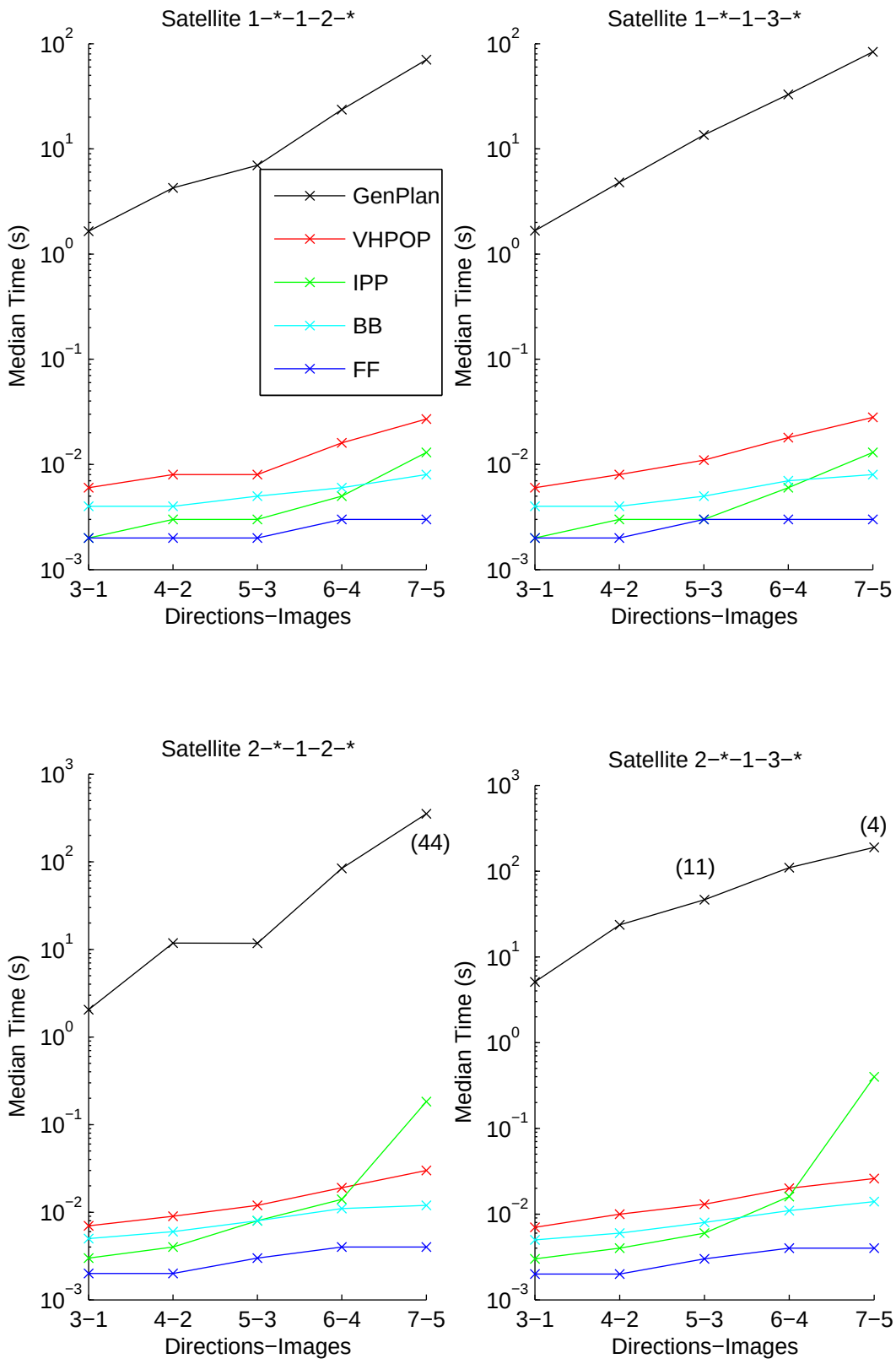



Figure 8.27: Efficiency results for `satellite`

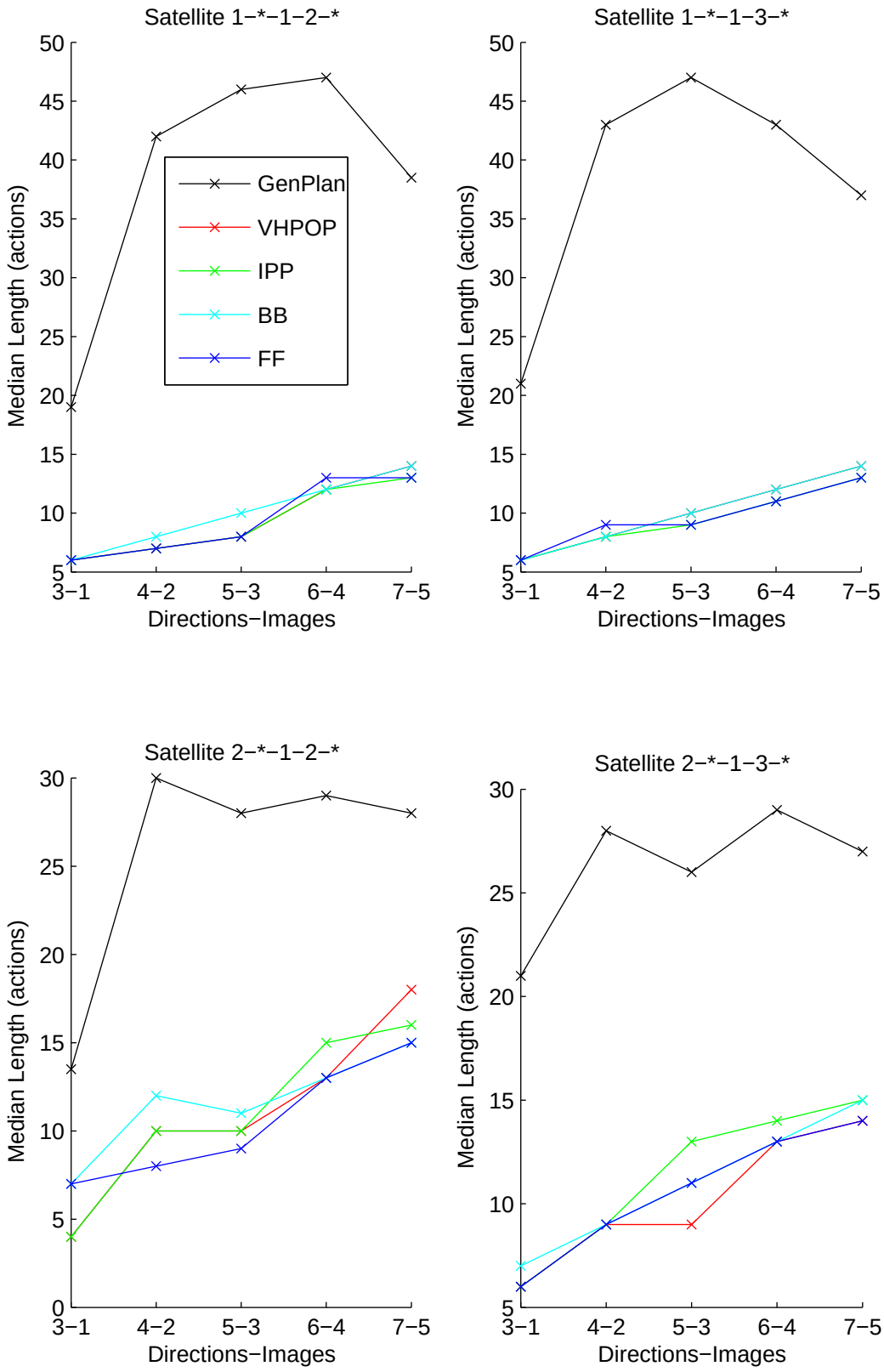



Figure 8.28: Plan quality results for `satellite`

this will be investigated by pooling both the random and IPC results into a single collection. However results from the following problem have been removed to avoid ceiling affects: pfile4.

- **Null Hypothesis** $H_{n0}$: The relationship of solution time between IPP and GENPLAN is linear for the `satellite` 1-*-1-2-* problems.
  **Alternative Hypothesis**: The relationship of solution time between IPP and GENPLAN contains a significant degree of non-linearity.

- **Null Hypothesis** $H_{n1}$: The relationship of solution time between IPP and GENPLAN is linear for the `satellite` 1-*-1-3-* problems.
  **Alternative Hypothesis**: The relationship of solution time between IPP and GENPLAN contains a significant degree of non-linearity.

- **Null Hypothesis** $H_{n2}$: The relationship of solution time between IPP and GENPLAN is linear for the `satellite` 2-*-1-2-* problems.
  **Alternative Hypothesis**: The relationship of solution time between IPP and GENPLAN contains a significant degree of non-linearity.

- **Null Hypothesis** $H_{n3}$: The relationship of solution time between IPP and GENPLAN is linear for the `satellite` 2-*-1-3-* problems.
  **Alternative Hypothesis**: The relationship of solution time between IPP and GENPLAN contains a significant degree of non-linearity.

- **Null Hypothesis** $H_{n4}$: The relationship of solution time between VHPOP and GENPLAN is linear.
  **Alternative Hypothesis**: The relationship of solution time between VHPOP and GENPLAN contains a significant degree of non-linearity.

- **Null Hypothesis** $H_{n5}$: The relationship of solution time between BB and GENPLAN is linear.
  **Alternative Hypothesis**: The relationship of solution time between BB and GENPLAN contains a significant degree of non-linearity.

- **Null Hypothesis** $H_{n6}$: The relationship of solution time between FF and GENPLAN is linear.
  **Alternative Hypothesis** The relationship of solution time between FF and GENPLAN contains a significant degree of non-linearity.

The discussion refers to both Table 8.23 and Figure 8.29.

- There is insufficient evidence to reject $H_{n0}$.
- There is insufficient evidence to reject $H_{n1}$.
- There is sufficient evidence to reject $H_{n2}$. There is a significant degree of non-linearity and due to the shape of the curve GENPLAN is scaling better than IPP for the 2-*-1-2-* problems.
- There is sufficient evidence to reject $H_{n3}$. There is a significant degree of non-linearity and due to the shape of the curve GENPLAN is scaling better than IPP for the 2-*-1-3-* problems.
- There is sufficient evidence to reject $H_{n4}$. There is a significant degree of non-linearity and due to the shape of the curve GENPLAN is scaling worse than VHPOP.
- There is sufficient evidence to reject $H_{n5}$. There is a significant degree of non-linearity and due to the shape of the curve GENPLAN is scaling worse than BB.
- There is sufficient evidence to reject $H_{n6}$. There is a significant degree of non-linearity and due to the shape of the curve GENPLAN is scaling worse than FF.

The non-linearity test has confirmed the GENPLAN does scale better than IPP on some of the `satellite` problems sets: 2-*-1-2-* and 2-*-1-3-*. Also that GENPLAN is scaling worse than VHPOP, BB, and FF for the `satellite` domain.

| Model - Efficiency | $R^2$ Linear | $R^2$ Non-linear | P-value |
|---|---|---|---|
| $IPP \sim GenPlan$ 1-*-1-2-* | 0.9931 | 0.9968 | 0.2671 |
| $IPP \sim GenPlan$ 1-*-1-3-* | 0.9913 | 0.9933 | 0.5217 |
| $IPP \sim GenPlan$ 2-*-1-2-* | 0.9705 | 0.9994 | 0.00944 |
| $IPP \sim GenPlan$ 2-*-1-3-* | 0.7495 | 0.9792 | 0.0424 |
| $VHPOP \sim GenPlan$ | 0.6527 | 0.7826 | 0.00249 |
| $BB \sim GenPlan$ | 0.5709 | 0.8067 | $<$0.0001 |
| $FF \sim GenPlan$ | 0.5309 | 0.7013 | 0.00384 |

Table 8.23: Statistical data for Figure 8.29

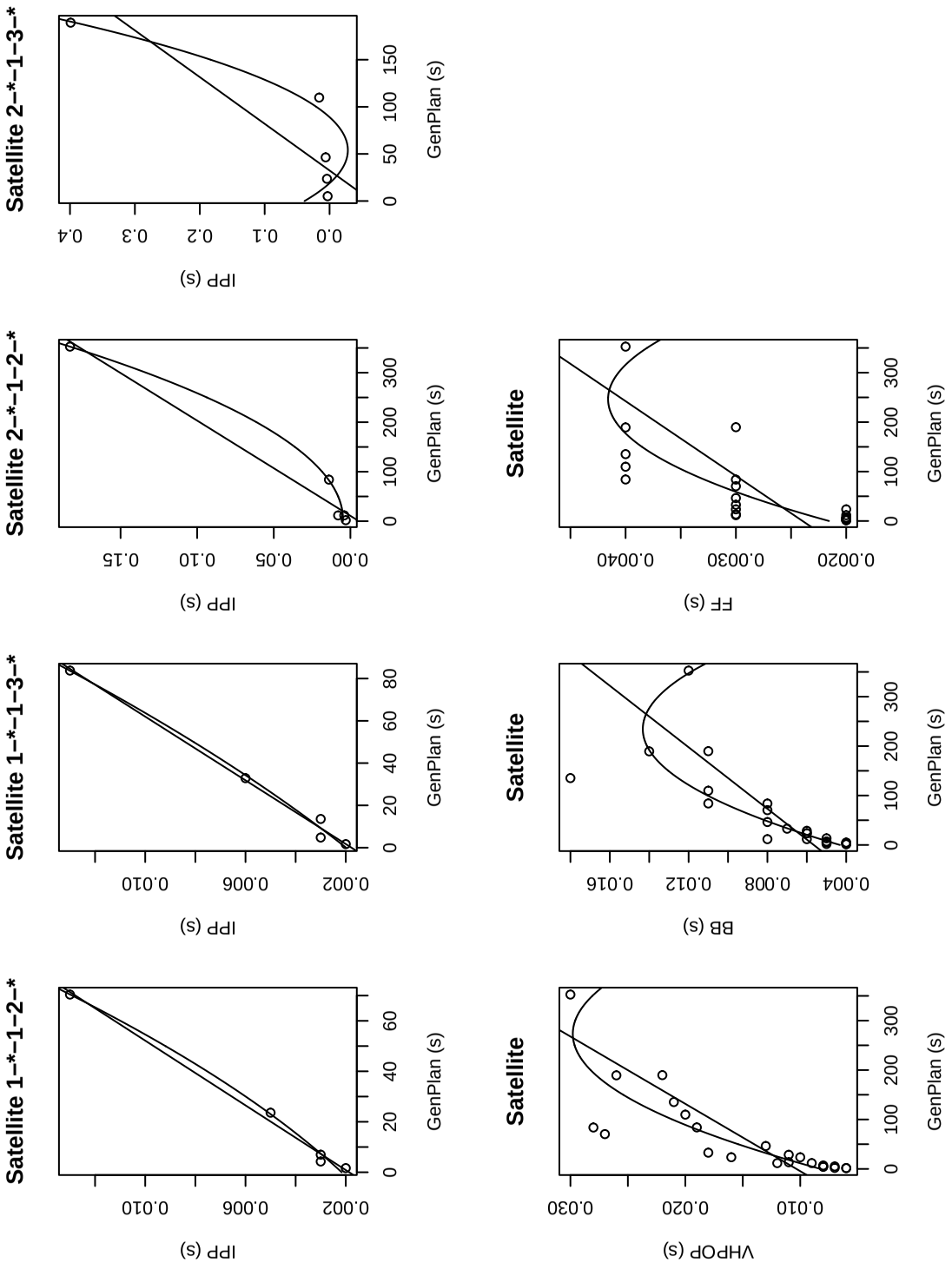



Figure 8.29: Efficiency Regression Plots for `satellite`

#### 8.2.8.3 Evaluation - Quality

Referring to Figure 8.28 and Table 8.22 it is not possible to conclude a linear relationship of plan quality between GENPLAN and the four test planners. The non-linearity test will be applied to the data. Again, the results from pfile4 are not included in the analysis.

- **Null Hypothesis** $H_{n0}$: The relationship of plan quality between VHPOP and GENPLAN is linear.
  **Alternative Hypothesis**: The relationship of plan quality between VHPOP and GENPLAN contains a significant degree of non-linearity.

- **Null Hypothesis** $H_{n1}$: The relationship of plan quality between IPP and GENPLAN is linear.
  **Alternative Hypothesis**: The relationship of plan quality between IPP and GENPLAN contains a significant degree of non-linearity.

- **Null Hypothesis** $H_{n2}$: The relationship of plan quality between BB and GENPLAN is linear.
  **Alternative Hypothesis**: The relationship of plan quality between BB and GENPLAN contains a significant degree of non-linearity.

- **Null Hypothesis** $H_{n3}$: The relationship of plan quality between FF and GENPLAN is linear.
  **Alternative Hypothesis** The relationship of plan quality between FF and GENPLAN contains a significant degree of non-linearity.

The discussion refers to both Table 8.24 and Figure 8.30.

- There is sufficient evidence to reject $H_0$. There is a significant degree of non-linearity and due to the shape of the curve GENPLAN is scaling worse than VHPOP.

- There is sufficient evidence to reject $H_1$. There is a significant degree of non-linearity and due to the shape of the curve GENPLAN is scaling worse than IPP.

- There is sufficient evidence to reject $H_2$. There is a significant degree of non-linearity and due to the shape of the curve GENPLAN is scaling worse than BB.

| Model - Quality | $R^2$ Linear | $R^2$ Non-linear | P-value |
|---|---|---|---|
| $VHPOP \sim GenPlan$ | 0.05383 | 0.3853 | 0.00371 |
| $IPP \sim GenPlan$ | 0.01209 | 0.3962 | 0.00193 |
| $BB \sim GenPlan$ | 0.03927 | 0.3309 | 0.00788 |
| $FF \sim GenPlan$ | 0.03213 | 0.2428 | 0.0286 |

Table 8.24: Statistical data for Figure 8.30

- There is sufficient evidence to reject $H_3$. There is a significant degree of non-linearity and due to the shape of the curve GENPLAN is scaling worse than FF.

The non-linearity test has confirmed that there is a significant non-linearity present in the length of plans produced by GENPLAN when compared with the other four test planners.

### 8.2.9 DriverLog

The `driverlog` domain is a more complicated transportation domain involving two transportation networks and drivers which are needed to drive trucks in order to transport objects to the right locations. The variables to vary are: the number of locations, the number of drivers, the number of trucks, and the number of packages. The problems range from 2 to 3 road junctions, with 1 to 2 drivers and 1 to 2 trucks. The number of packages was varied from 1 to 6. The naming scheme used to describe the problems is: number of road nodes, number of path nodes, number of drivers, number of trucks, number of packages, and the number of object goals. Goal locations are given for drivers and trucks creating multiple one action local minima. All the problems were created using the random problem generator used for IPC 3. The IPC random problem generator does not always include a goal for all packages, that is why the number of goals is included in the naming. A selection of problems from IPC 3 are also used.

- Let $O$ be number of objects, $T$ the number of trucks, $L$ be the number of nodes, including path and road nodes, and $D$ the number of drivers.
  - number of `load_truck(?o ?t ?l)` actions $= O*T*L$
  - number of `unload_truck(?o ?t ?l)` actions $= O*T*L$
  - number of `board_truck(?d ?t ?l)` actions $= D*T*L$

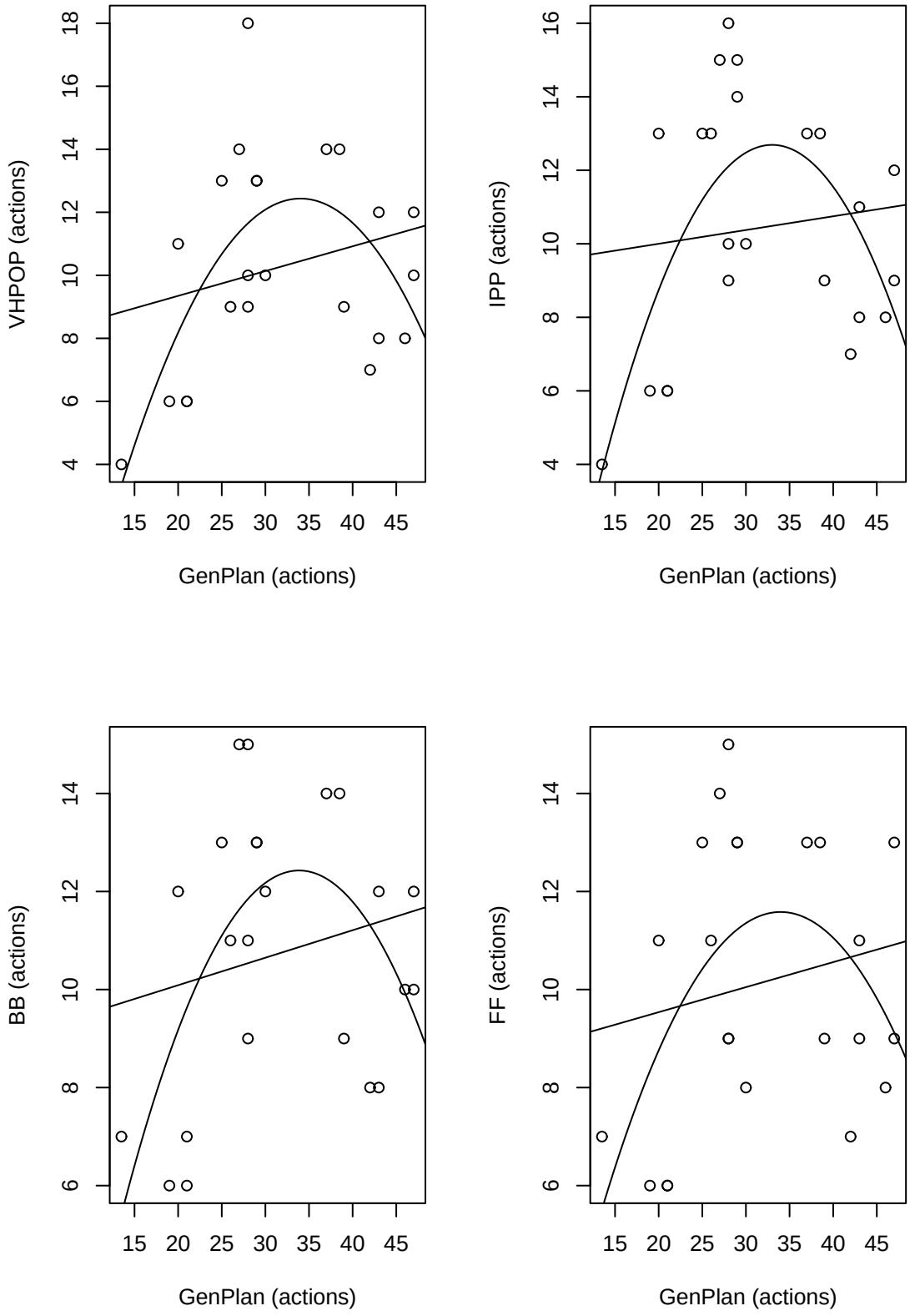



Figure 8.30: Regression plots for plan quality for `satellite`

| Problem | GenPlan | VHPOP | IPP | BB | FF |
|---|---|---|---|---|---|
| pfile1 (3-2-2-2-2-2) | 60.9 | 0.012 | 0.005 | 0.011 | 0.003 |
| pfile2 (3-4-2-2-3-3) | 498.18 (89) | 3.075 | 0.043 | 0.151 | 0.008 |
| pfile3 (3-3-2-2-4-3) | 497.58 (56) | 0.025 | 0.015 | 0.064 | 0.004 |
| pfile4 (3-4-3-2-4-4) | 498.29 (100) | 10.054 | 0.096 | 0.126 | 0.009 |
| pfile5 (3-3-3-2-5-5) | 498.08 (89) | 0.211 | 1.043 | 0.199 | 0.006 |
| pfile6 (3-2-3-3-5-5) | 223.11 (29) | 0.04 | 0.019 | 0.055 | 0.075 |
| pfile7 (3-3-3-3-6-6) | 498.90 (98) | 0.093 | 0.122 | 0.155 | 0.006 |
| pfile8 (3-3-3-3-7-7) | 498.48 (100) | 1.749 | 2.592 | 0.285 | 0.019 |

Table 8.25: Efficiency results for `driverlog` on selected IPC 3 instances

  - number of `disembark_truck(?d ?t ?l)` actions $= D*T*L$
  - number of `drive_truck(?t ?l ?l ?d)` actions $= T*L*(L-1)*D$
  - number of `walk(?d ?l ?l)` actions $= D*L*(L-1)$

- Action space: 40-372 actions

#### 8.2.9.1 Results

Referring to Figure 8.31, GENPLAN appears to be scaling poorly in comparison with the four test planners. FF presents the best raw efficiency results and scaling behaviour. Table 8.25 presents results from a handful of IPC3 problems.

As already mentioned for the `miconic` domain GENPLAN's efficiency will suffer slightly due to the way in which actions are created independent of a particular problem. This domain is also difficult as locations within either subgraph of locations are not fully connected (the paths and the roads). Finally, the domain could be typed better so that actions are not created which involve attempting to move trucks along paths. This action space issue of domain engineering is returned to in Chapter 9.

Referring to Figure 8.32, GENPLAN is producing a mixture plans of lower quality and of quite similar quality when compared to the four test planners. It is difficult to pick a winner from the four test planners for this domain. Table 8.26 presents results from a handful of IPC3 problems.

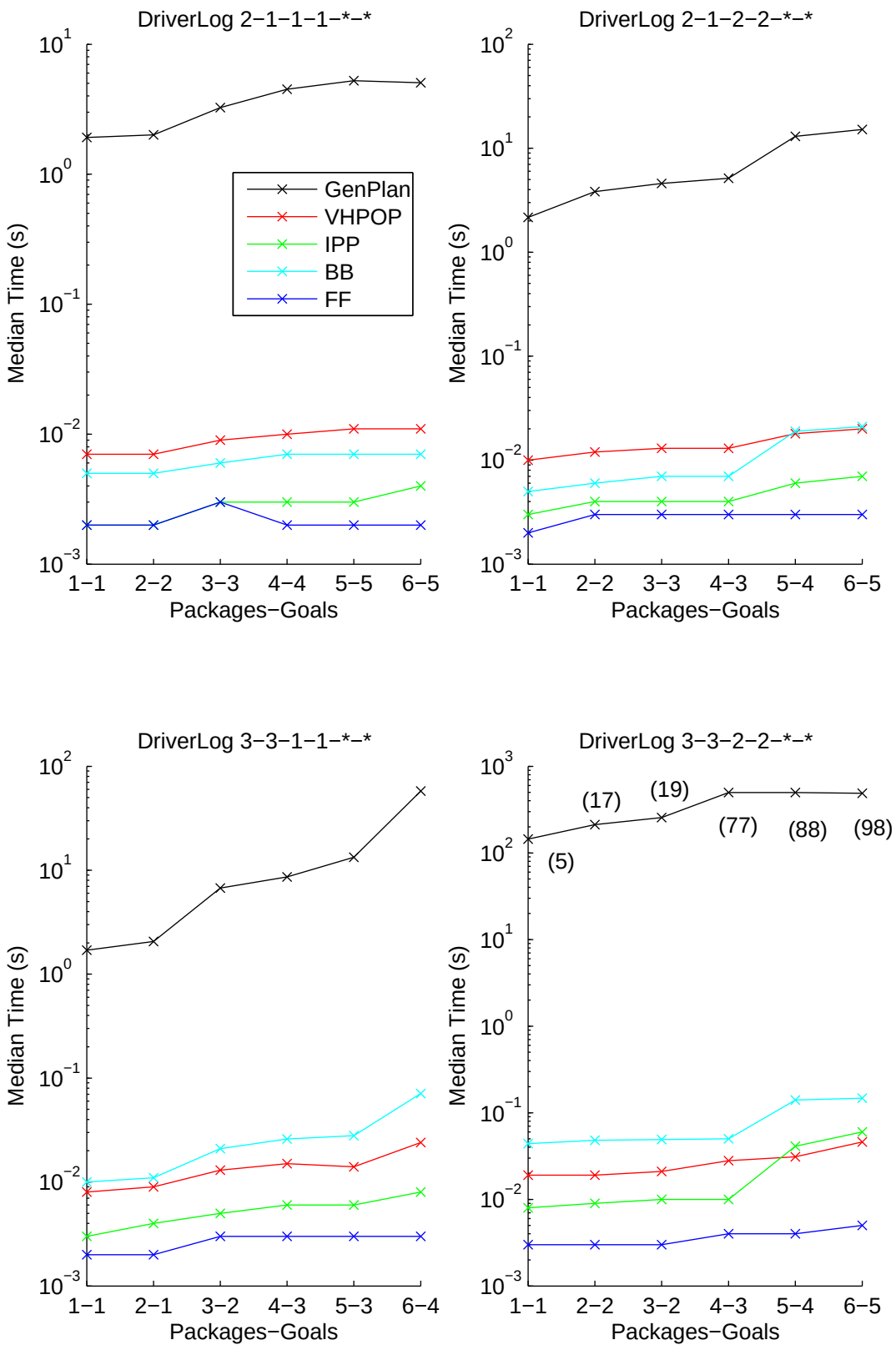



Figure 8.31: Efficiency results for `driverlog`

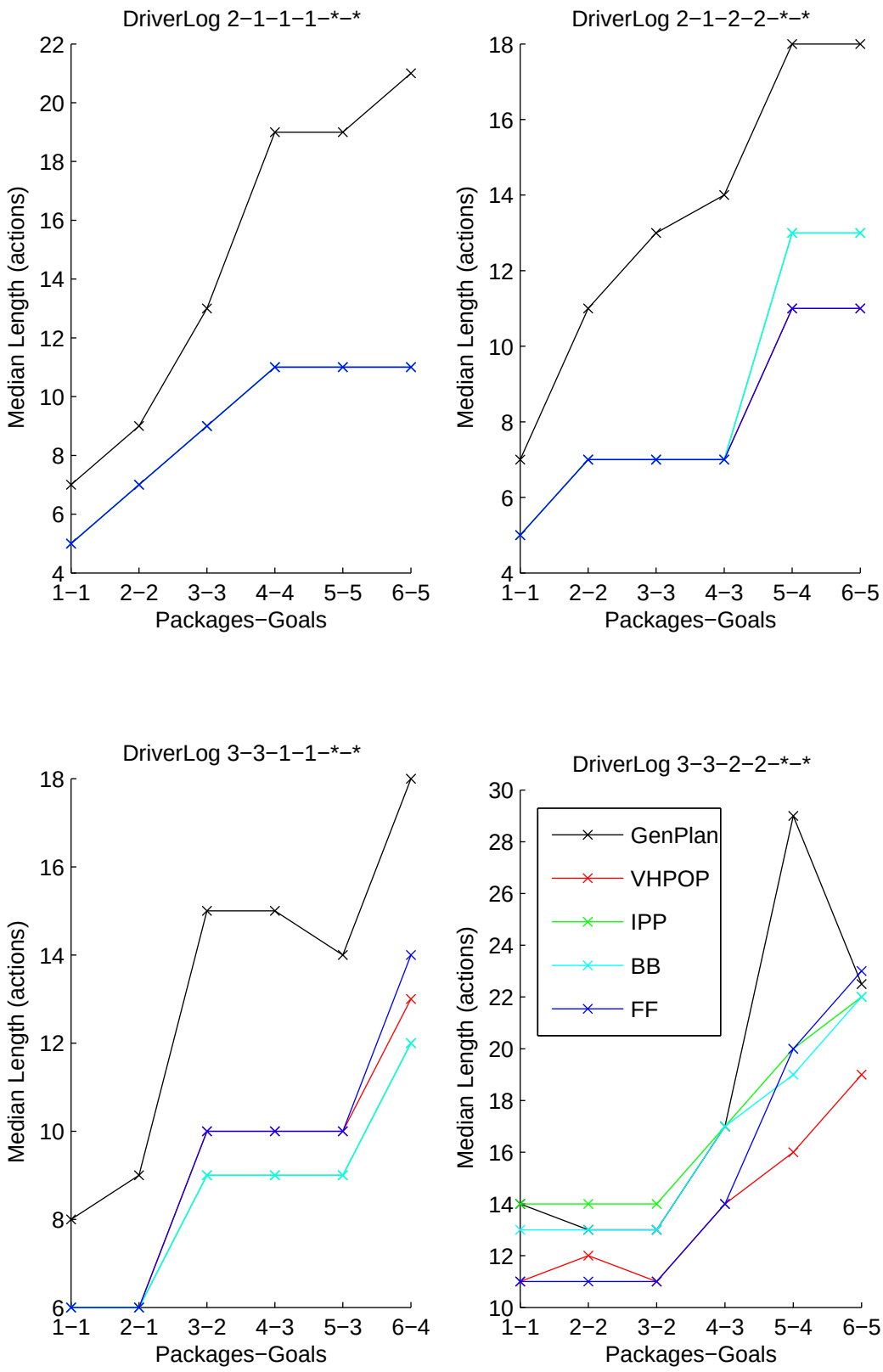



Figure 8.32: Plan Quality results for `driverlog`

| Problem | GenPlan | VHPOP | IPP | BB | FF |
|---|---|---|---|---|---|
| pfile1 (3-2-2-2-2-2) | 10 | 7 | 8 | 8 | 8 |
| pfile2 (3-4-2-2-3-3) | 20 | 21 | 19 | 19 | 23 |
| pfile3 (3-3-2-2-4-3) | 16 | 13 | 12 | 17 | 12 |
| pfile4 (3-4-3-2-4-4) | - | 16 | 18 | 18 | 20 |
| pfile5 (3-3-3-2-5-5) | 22 | 18 | 19 | 19 | 23 |
| pfile6 (3-2-3-3-5-5) | 14 | 11 | 11 | 11 | 13 |
| pfile7 (3-3-3-3-6-6) | 14 | 15 | 18 | 18 | 17 |
| pfile8 (3-3-3-3-7-7) | - | 25 | 25 | 25 | 29 |

Table 8.26: Plan quality results for `driverlog` on selected IPC 3 instances

#### 8.2.9.2 Evaluation - Efficiency

The previous section speculated that GENPLAN is scaling worse than the four test planners. The analysis includes results from Figure 8.31 and Table 8.25 but with results from the following problems removed in order to protect against ceiling effects: 3-3-2-2-5-4, 3-3-2-2-6-5, pfile2, pfile4, pfile5, pfile7, pfile8.

- **Null Hypothesis** $H_{n0}$: The relationship of solution time between VHPOP and GENPLAN is linear.
  **Alternative Hypothesis**: The relationship of solution time between VHPOP and GENPLAN contains a significant degree of non-linearity.

- **Null Hypothesis** $H_{n1}$: The relationship of solution time between IPP and GENPLAN is linear.
  **Alternative Hypothesis**: The relationship of solution time between IPP and GENPLAN contains a significant degree of non-linearity.

- **Null Hypothesis** $H_{n2}$: The relationship of solution time between BB and GENPLAN is linear.
  **Alternative Hypothesis**: The relationship of solution time between BB and GENPLAN contains a significant degree of non-linearity.

- **Null Hypothesis** $H_{n3}$: The relationship of solution time between FF and GENPLAN is linear.

| Model - Efficiency | $R^2$ Linear | $R^2$ Non-linear | P-value |
|---|---|---|---|
| $VHPOP \sim GenPlan$ | 0.4964 | 0.5902 | 0.0126 |
| $IPP \sim GenPlan$ | 0.5902 | 0.7228 | 0.00373 |
| $BB \sim GenPlan$ | 0.5631 | 0.6963 | 0.00514 |
| $FF \sim GenPlan$ | 0.05211 | 0.2301 | 0.0344 |

Table 8.27: Statistical data for Figure 8.33

**Alternative Hypothesis** The relationship of solution time between FF and GENPLAN contains a significant degree of non-linearity.

The discussion refers to both Table 8.27 and Figure 8.33.

- There is sufficient evidence to reject $H_{n0}$. There is a significant degree of non-linearity and due to the shape of the curve GENPLAN is scaling worse than VHPOP.
- There is sufficient evidence to reject $H_{n1}$. There is a significant degree of non-linearity and due to the shape of the curve GENPLAN is scaling worse than IPP.
- There is sufficient evidence to reject $H_{n2}$. There is a significant degree of non-linearity and due to the shape of the curve GENPLAN is scaling worse than BB.
- There is sufficient evidence to reject $H_{n3}$. There is a significant degree of non-linearity and due to the shape of the curve GENPLAN is scaling worse than FF.

The non-linearity test has confirmed that GENPLAN is scaling worse than the four test planners.

#### 8.2.9.3 Evaluation - Quality

Referring to Figure 8.32 and Table 8.26 it difficult to confirm a linear relationship between GENPLAN and four other planners. The results included in Figure 8.34 are from Figure 8.32 and Table 8.26 but with results from the following problems removed in order to protect against ceiling effects: 3-3-2-2-5-4, 3-3-2-2-6-5, pfile2, pfile4, pfile5, pfile7, pfile8. GENPLAN did produce plans of quite variable quality for this domain. The non-linearity test reported that there is insufficient evidence to claim a significant non-linearity in the data for each comparison between GENPLAN and another test planner.

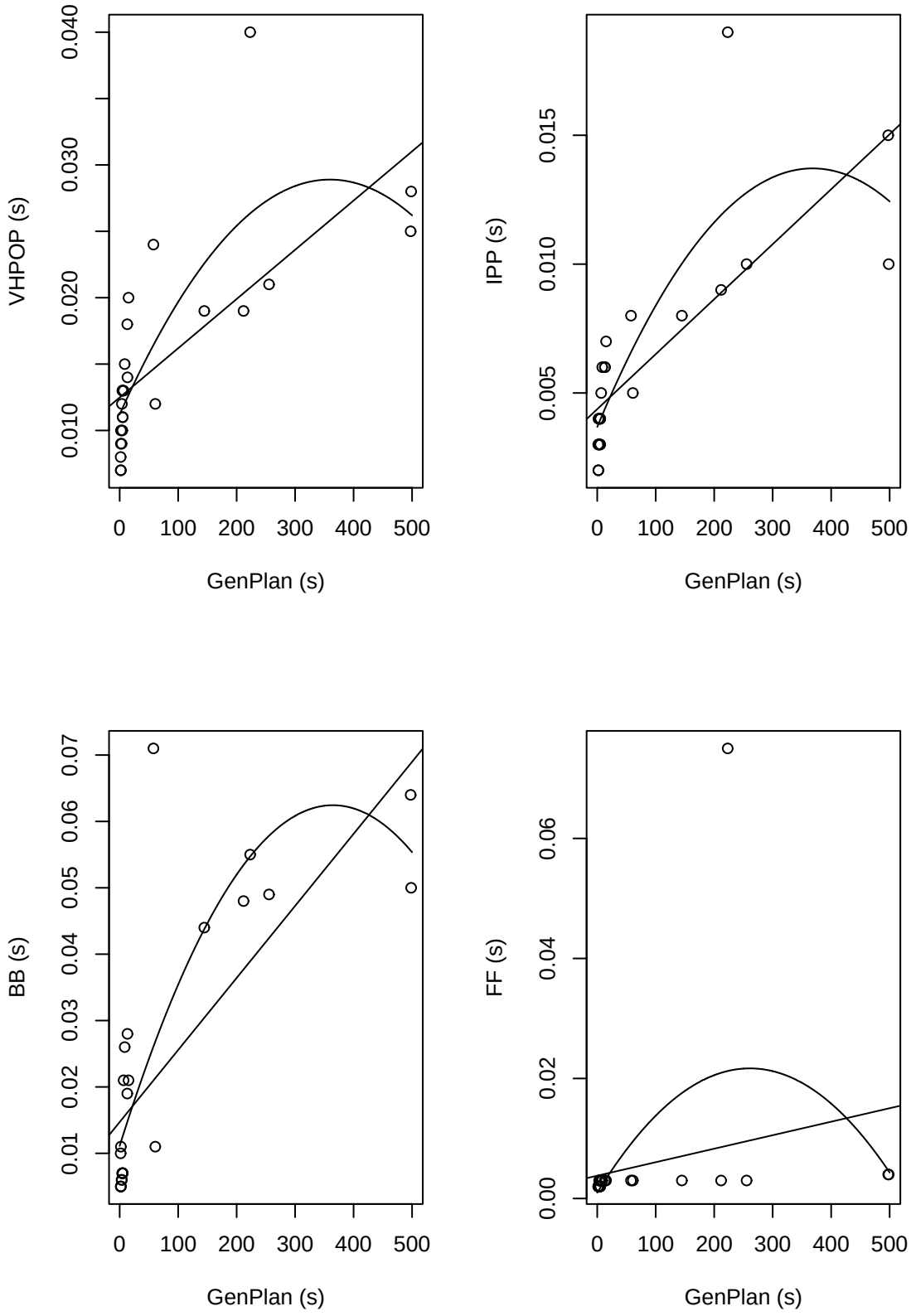



Figure 8.33: Efficiency Regression Plots for `driverlog`

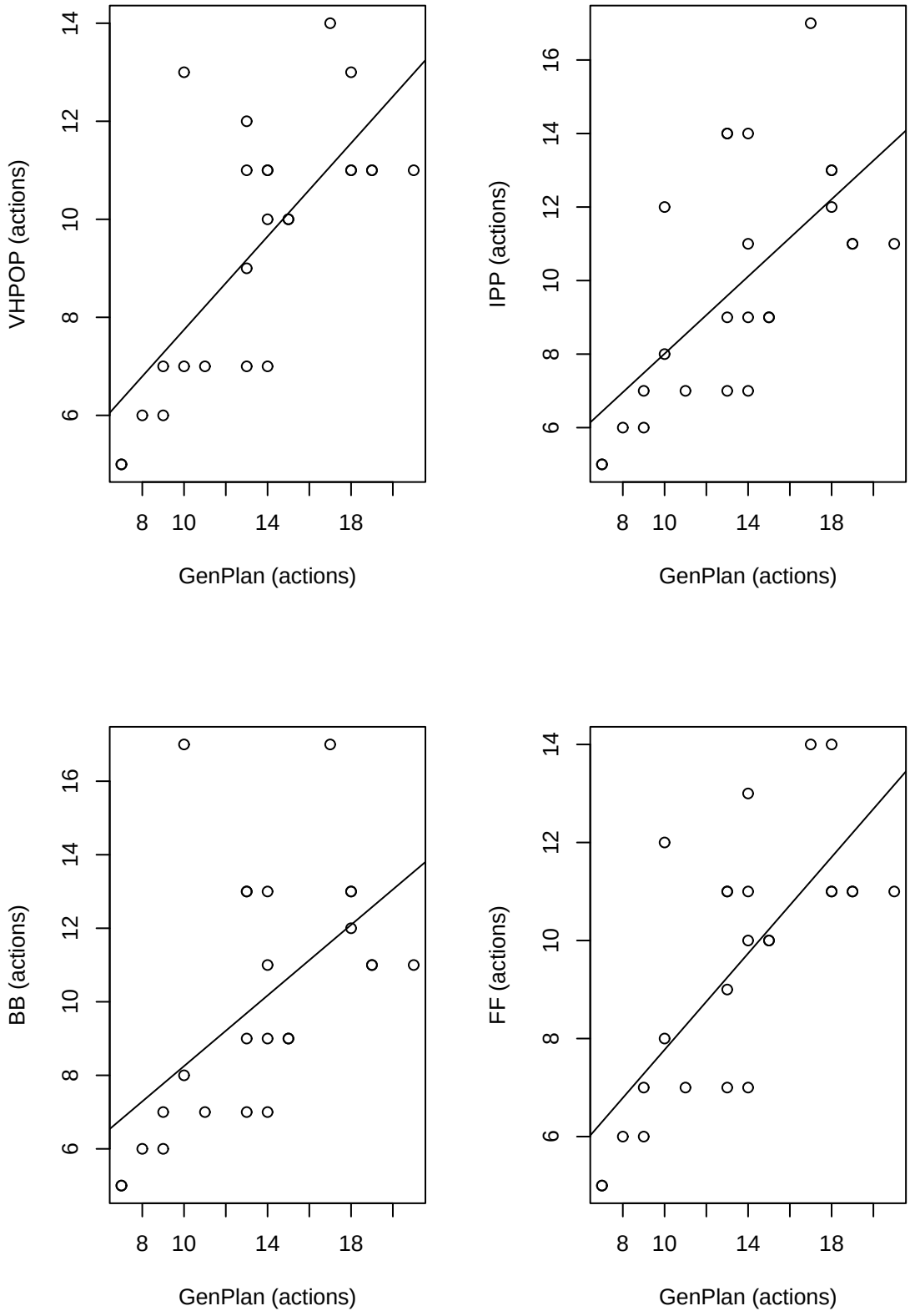



Figure 8.34: Regression plots for plan quality for `driverlog`

| Model - Quality | $R^2$ Linear |
|---|---|
| $VHPOP \sim GenPlan$ | 0.5207 |
| $IPP \sim GenPlan$ | 0.4171 |
| $BB \sim GenPlan$ | 0.3183 |
| $FF \sim GenPlan$ | 0.5416 |

Table 8.28: Statistical data for Figure 8.34

## 8.3 Chapter Summary

This chapter presents experimental results of running GENPLAN on a variety of planning domains: including standard and non-standard benchmark domains. The selection of domains was performed rigorously by selecting domains in relation to known search metrics that attempt to describe the landscape of the search-space and the subsequent difficulty of the domain. Table 8.29 summarises the characteristics of the domains.

Table 8.30 summarises the efficiency evaluation of GENPLAN. The GENPLAN approach is basic in order facilitate an exploration of EAs applied to domain independent planning and so these results demonstrate initial but thorough results. In comparison to the other planners GENPLAN performed best against IPP and worst against FF. A couple of positive results are also reported in comparison with VHPOP and BB. This suggests that GENPLAN can plan but has some way to go to be competitive with the better modern planners. Also because the evaluation was performed after the results were collected, any future analysis should make a more careful choice of problems, in terms of the non-linearity test, so that more conclusive results can be generated.

Table 8.31 summarises the plan quality evaluation of GENPLAN. Those domains where the results were inconclusive have been left out of the table. Generally, GENPLAN produces the worst raw quality plans of all the planners. The quality of plan produced by GENPLAN is normally a constant factor longer than that of a plan produced by a competition planner. The only exception to this was the `satellite` domain where GENPLAN's plans appear to be much worse than those of the other test planners. However, this is not taking into account GENPLAN's ability to optimise its own plans [Westerberg and Levine, 2001b, Westerberg and Levine, 2001c], though this additional step would include additional solution time. An additional partial fix to this problem would to upgrade the plan rewriter component with a greedy loop removal

| Domain | Dead-ends | Number of Actions | Max Local Minima | Min Bench Size |
|---|---|---|---|---|
| Maxones | harmless | 1 | 0 | 1 |
| TTD | harmless | 2 | 0 | 1 |
| Briefcase | undirected | 3 | 0 | 3 |
| **Gripper** | undirected | 3 | 1 | 3 |
| Blocksworld-3ops | undirected | 3 | $< c$* | $< c$ |
| Numeric Mystery | unrecognised | 3 | 0 | 3 |
| **Miconic-STRIPS** | harmless | 4 | 0 | 3 |
| **Satellite** | harmless | 5 | 1* | 3 |
| **DriverLog** | undirected | 6 | 1* | 4 |

Table 8.29: The selected domains with adjusted local minima values to reflect problem sets used

algorithm which could quickly remove the worst loops in a plan.

Despite the poor raw efficiency results of GENPLAN, the results in this chapter have shown that GENPLAN can achieve better scaling behaviour than other planners for some problem sets. It is also clear that GENPLAN scales badly for the harder domains in the suite of domains, in particular `driverlog`. However planning is a general problem, for which not all facets have been represented by the current range of domains. With further exploration of planning, a planning problem may exist that an EA-based approach is well suited to when compared to other search-based planners. A particular strength of the approach is the simulation model used, this would allow GENPLAN to plan for domains which cannot currently be expressed in PDDL. Apollo 13 was a domain that I defined in an attempt to achieve this [Westerberg and Levine, 2003]. The search issues encountered in this chapter are highlighted and explored in the next chapter.

| Domain | VHPOP | IPP | BB | FF |
|---|---|---|---|---|
| Maxones | worse | worse | worse | worse |
| Maxones-1600 | better | better | worse | better |
| TTD 10 Cities | inc. | better | inc. | inc. |
| TTD 20 Cities | inc. | better | inc. | worse |
| TTD 30 Cities | inc. | inc. | worse | worse |
| TTD 40 Cities | better | better | inc. | inc. |
| Briefcase 6 Cities | inc. | better | inc. | inc. |
| Briefcase 7 Cities | inc. | better | inc. | inc. |
| Briefcase 8 Cities | inc. | better | worse | inc. |
| Briefcase 9 Cities | inc. | better | inc. | inc. |
| Gripper | better | better | better | worse |
| Blocksworld-3ops | inc. | worse | worse | worse |
| Numeric Mystery | - | - | - | worse |
| Miconic-STRIPS 6 Floors | worse | better | inc. | worse |
| Satellite | worse | better | worse | worse |
| DriverLog | worse | worse | worse | worse |

Table 8.30: A quick summary of the scaling evaluation of GENPLAN

| Domain | VHPOP | IPP | BB | FF |
|---|---|---|---|---|
| Blocksworld-3ops | better | inc. | inc. | inc. |
| Satellite | worse | worse | worse | worse |

Table 8.31: A quick summary of the plan quality evaluation of GENPLAN

# Chapter 9

# Issues with the GENPLAN Approach

This chapter identifies known search issues with the GENPLAN approach. The issues of GENPLAN can be divided into two main areas, those which derive from the planning problem, and those that have been already identified in the EA field but which manifest themselves for this application of EAs. Each issue is summarised and some will include experimental results highlighting the issue. Experiments will involve domains and problems from the previous chapter and new hand coded pathological problems in order to demonstrate a particular search issue.

This chapter is vitally important as it highlights the difficulties with the GENPLAN approach to fully automated planning. This chapter is intended as a guide should anyone else attempt a similar approach. It is important when proposing a new approach to attempt to identify strengths and weaknesses of the approach. It is too often the case that there is little understanding of what is happening during search and when search might fail. The issues of search that feature in this work have appeared in many areas of AI, including fully automated planning, though some are specific to EAs.

The main planning search issues highlighted for this chapter are:

- **Distraction:** This issue is the impact that ever increasing action spaces have on the efficiency of GENPLAN.

- **Benches:** An area in the search space where for each point in that area the algorithm receives the same fitness value. This issue relates the effect of ever increasingly sized benches in the search space with the efficiency of GENPLAN.

- **Local Minima:** A point or set of points where the current fitness value needs to be decreased in order to reach an area of the search space where a solution can

be found. The effect of variable sized local minima and of multiple small local minima are examined in relation to the efficiency of GENPLAN.

- **Resources:** A resource forms an additional constraint on the creation of any plan to solve a particular problem. Two forms of resources are examined: consumable resources, and resources involving restricted carrying capacity. The issue of multiple carrying resources is mainly a distraction issue.

The main EA search issues highlighted in this chapter are:

- **Diversity:** This is a measure of how distinct individual candidates are from one another and also how distinct the solutions they create are.
- **Bloat:** This term is used to describe the often massive growth in length of candidates during a run. A side effect of which is poor quality plans.
- **Crossover:** At the heart of most EAs is crossover and is an operator that has undergone the most scrutiny in the EA community, whether it actually performs as expected is an open issue.

## 9.1 Experimental Methodology

The experimental methodology for this chapter is to use experimentation to highlight particular features of search against the efficiency of GENPLAN. The small scale experiments are used to present the behaviour of GENPLAN. The experiments are a series of observational experiments, designed to discover points of failure of the algorithm, when a particular search issue becomes too difficult to overcome. Each experiment is considered separately with no relation to other experiments in the chapter. In the final experiment of the chapter a manipulation experiment is used to compare the behaviour of two crossover operators.

### 9.1.1 Choose Goals

The primary purpose of the experiments in this chapter are to demonstrate the effects of different search issues on the behaviour of GENPLAN. This will be achieved by performing limited experiments using domains and problems related specifically to the issue being discussed. The final experiment in this chapter will attempt to show a significant difference between two different crossover operators on the efficiency of GENPLAN.

### 9.1.2 Choose Metrics

For the most part, the standard metrics for both efficiency and quality presented in Chapter 6 will be used, though this chapter is primarily concerned with efficiency. Two additional metrics are used for this chapter: the first metric measures the diversity of different actions at the end of a run and the second records the growth of candidates during a run of GENPLAN. They are fully explained in relation to each particular experiment.

### 9.1.3 Choose Problems

For this chapter, domains and problems are chosen to demonstrate particular issues of the algorithm and to achieve some generality for the manipulation experiment. The choice is again broken down to choosing domains and the problems to those domains.

#### 9.1.3.1 Choosing Domains

The domains used in this chapter are:

- `driverlog`: This domain is used to demonstrate two different search issues: distraction and multiple small local minima.
- `ttd_conn`: This domain is a variant of `TTD`. The difference is that there are explicit connections between cities rather than assuming all cities are connected to one another. This domain is used to demonstrate the effect of benches on the efficiency of GENPLAN search.
- `blocksworld-3ops`: A domain of key importance as it can involve problems with arbitrarily sized local minima.
- `satellite`: This domain is also used to demonstrate the effect of multiple small local minima.
- `mystery`: The only domain that involves consumable resources which can lead to unrecognised dead-ends in the search space.
- `gripper, briefcase`: Together these domains will compare infinite carrying resources with limited carrying capacity.
- `maxones`: This domain will be used to demonstrate the issue of action diversity.

- The four parameterisation domains used for Chapter 7 will also be used: `maxones`, `briefcase`, `blocksworld-3ops`, `satellite`. These domains are chosen to demonstrate bloat and to compare two crossover operators.

#### 9.1.3.2 Choosing Problems to the domains

For the most part the choice of problems is different for this chapter than with the other chapters. The problems are chosen to highlight specific features of search, so instead of relying on random problem generators to create these features by chance it is simple to hand-code problems to clearly demonstrate the effect of different search issues. These problems are pathological in nature and do not always represent the general case. The choice and selection of problems will be clearly marked with each experiment. For the experiments involving crossover and bloat the parameterisation problems are re-used, as they have an adequate range of difficulty for each domain.

### 9.1.4 Design of Experiments

For most of the issues an observational experiment is sufficient in order to demonstrate the issue. The experiments for this chapter are very much in the small scale and serve only to highlight key issues. In addition, one manipulation experiment is performed in order to compare two types of crossover.

### 9.1.5 Running the Experiment

All experiments were run using the parameterisation decided on in Chapter 7. All experiments were run in accordance with Chapter 6.

### 9.1.6 Presenting the Results

The standard plots and tables are used for this chapter. The EA section of the chapter makes use of additional presentation devices and they will be described at that point.

## 9.2 Planning Search Issues

The planning issues relate to those created by the nature of the search space, and the interaction of GENPLAN with the search space. As shown in Chapter 2, planning is a hard problem and as such comes with its own well studied complexity results.

However little effort is made to relate the search algorithm against the characteristics of the planning domains. The one exception to this is the work done by Jörg Hoffman with FF [Hoffmann, 2003]. The issues that will be discussed are: distraction, benches, local minima, dead-ends, and resources.

### 9.2.1 Distraction (Action Space)

The action space is the total number of instantiated actions that can be made given a particular domain and problem pairing. In the previous chapter, formulae were given to calculate the number of instantiated actions for each domain. The action space has been used as a metric for measuring the difficulty of particular problems [Long et al., 2000b, Long and Fox, 2003a].

The issue that large action spaces create is distraction. A particular domain and problem pairing will contain actions that are not needed in finding a solution to the problem. The larger the action space, the larger the size of the search space, thus making the problem more difficult. The same is true for GENPLAN as problems with larger actions spaces are harder to solve with all other factors being equal.

Exact action spaces can be calculated for a domain problem pairing by knowing the number of objects in the problem. How a domain is modelled can have a large effect on the resulting size of the action space. However, as was identified in the previous chapter, additional actions can be ruled out with a better understanding of the mechanics of the domain.

#### 9.2.1.1 Reduced Action Space through better Domain Modelling: Example 1

The growth of the action space for the IPC 3 version of `driverlog` can be decreased by improving the type system used within the domain. The goal of which will be to reduce the number of actions per problem. The current problem with the domain is that actions can be created that involve trucks at locations that are never possible, due to the type model used. An improved typing system can ensure that actions are not created that involve trucks attempting to traverse path nodes, or involving trucks at path nodes. The original type declaration and one action from `driverlog` are given below:

```
(:types location locatable - object
        driver truck obj - locatable)

(:action DRIVE-TRUCK
```

```
  :parameters (?truck - truck ?loc-from - location
               ?loc-to - location ?driver - driver)
  :precondition
   (and (at ?truck ?loc-from)
        (driving ?driver ?truck) (link ?loc-from ?loc-to))
  :effect
   (and (not (at ?truck ?loc-from)) (at ?truck ?loc-to)))
```

This specification of DRIVE-TRUCK allows actions that will attempt to drive trucks between any location irrespective of whether there is an actual "link" between the locations. In the domain, only roads are connected by links. The problem can be fixed by making a distinction between path nodes and road nodes. This can be done by subtyping the location type. The new type system and the updated version of DRIVE-TRUCK are given below:

```
(:types location locatable - object
        road path - location
        driver truck obj - locatable)

(:action DRIVE-TRUCK
  :parameters (?truck - truck ?loc-from - road
               ?loc-to - road ?driver - driver)
  :precondition
   (and (at ?truck ?loc-from) (driving ?driver ?truck)
        (link ?loc-from ?loc-to))
  :effect
   (and (not (at ?truck ?loc-from)) (at ?truck ?loc-to)))
```

The other operators in the domain are updated appropriately. The new types ensure that trucks are attempted to be driven only between road nodes. This reduces the action space for the domain. This also has the additional benefit of putting more actions that are likely to succeed in the action space. This form of type analysis could be automated using a domain analysis tool like TIM [Fox and Long, 1998].

- Let $O$ be number of objects, $T$ the number of trucks, $L$ be the number of nodes, including $P$ path and $R$ road nodes, and $D$ the number of drivers.

  - number of `load_truck(?o ?t ?l)` actions $= O * T * R$
  - number of `unload_truck(?o ?t ?l)` actions $= O * T * R$
  - number of `board_truck(?d ?t ?l)` actions $= D * T * R$
  - number of `disembark_truck(?d ?t ?l)` actions $= D * T * R$
  - number of `drive_truck(?t ?l1 ?l2 ?d)` actions $= T * R * (R-1) * D$
  - number of `walk(?d ?l1 ?l2)` actions $= D * L * (L-1)$

- Range of actions: 16 - 180 compared with 40 - 372 for the original version of the domain using the problems from Chapter 8.

Figure 9.1 presents results comparing the two domains from a subset of problems from Chapter 8. The problems come from the hardest set of the random problems, 3-3-2-2-*, * standing for the number of packages. No other changes have been made except the specification of domain. This enables GENPLAN to solve larger instances from the domain at the cost of a slight decrease in plan quality. The original results from Chapter 8 are incorporated into the figure.

#### 9.2.1.2 Reducing the number of actions through Domain Modelling: Example 2

As further examples of the effects of domain modelling on the size of the action space a series of examples are taken from different variations of `blocksworld`. The `blocksworld-4ops` domains is a variant in which a robot arm is explicitly modelled. The `blocksworld-2ops` domains consists of two actions: one to move blocks between towers and another to place a blocks on the table. Each version of the domain produces different sized action spaces for the same problem. For all three domains let $B$ be the number of blocks.

- `blocksworld-4ops`
  - number of `pickup(?b)` actions $= B$
  - number of `putdown(?b)` actions $= B$
  - number of `stack(?b1 ?b2)` actions $= B * (B-1)$
  - number of `unstack(?b1 ?b2)` actions $= B * (B-1)$
- Action space: 6 blocks = 72, 9 blocks = 162
- `blocksworld-3ops`

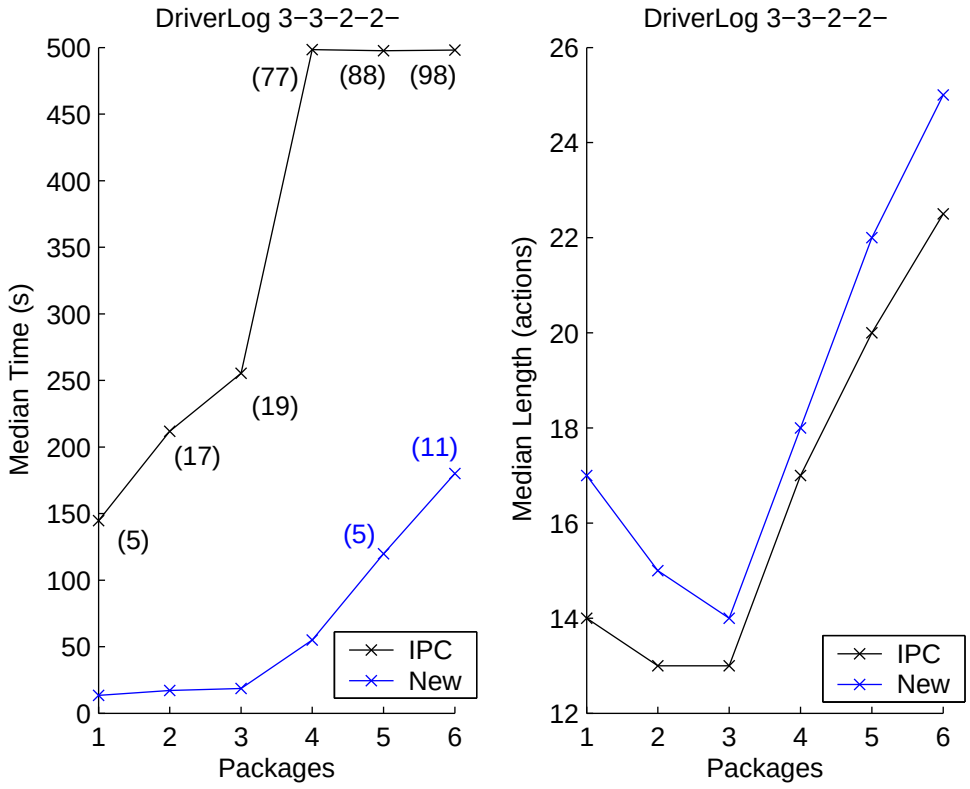



Figure 9.1: Presenting the efficiency of GENPLAN and quality of plans produced when using the original specification of `driverlog` and the improved type model version.

  - number of `move-b-to-b(?b1 ?b2 ?b3)` actions $= B*(B-1)*(B-2)$
  - number of `move-b-to-t(?b1 ?b2)` actions $= B*(B-1)$
  - number of `move-t-to-b(?b1 ?b2)` actions $= B*(B-1)$

- Action space: 6 blocks = 180, 9 blocks = 648

- `blocksworld-2ops`
  - number of `newtower(?b)` actions $= B$
  - number of `putin(?b1 ?b2)` actions $= B*(B-1)$

- Action space: 6 blocks = 36, 9 blocks = 81

A direct experimental comparison is not fully justified across all three domains, as `blocksworld-4ops` explicitly models the robot arm, which effectively doubles the length of plan that needs to be found to solve a problem. Also the search space is slightly harder for GENPLAN due to the extra interaction between actions, as it now takes two steps to move a block. But this simply adds to the argument that the encoding of domain requires careful consideration and can affect the search space. I performed a quick experiment using the six block problems and found that `blocksworld-4ops` was much more difficult than the other two versions of the domain. The `blocksworld-2ops` domain allowed for more efficient solution of most problems when compared with the `blocksworld-3ops` domain.

All three domains produce variously different sized action spaces for the same number of blocks. The `blocksworld-2ops` domain includes a partial solution to reducing the search space of planning. In the domain representation, local variables are required in order to identify blocks which sit on other blocks. The first version of GENPLAN, used in [Westerberg and Levine, 2000], relied on proceduralised versions of domains, which make looking up variables quite easy. The current version of GENPLAN relies on PDDL. An extension to PDDL to include local variables is discussed and explored in Chapter 10, with a view to reducing the action space and making a more friendly search space.

#### 9.2.1.3 Reducing the number of actions with Domain Analysis

For some domains, there are additional actions that should be ignored as they can never be executed, but are hard to find requiring an automatic analysis of domains.

As discussed in the previous chapter, the domains `miconic`, `mystery`, `driverlog` can all feature actions that are created, through permuting all possible operands to the operator, that will never succeed.

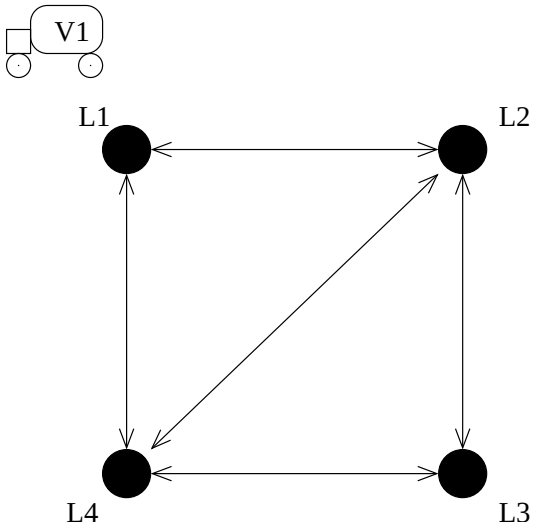



Figure 9.2: An example of a `mystery` map from which non-executable actions can be created

An example is given from the `mystery` domain, see Figure 9.2. Any action that attempts to move the vehicle between locations `L1` and `L3` will never succeed. This is a specific illustration of the type of action that arises that would not feature in a solution plan, or need to be searched over. One potential solution to this would be to perform reachability analysis using a planning graph [Blum and Furst, 1997], as during the initial graph construction phase a reduced list of all possible actions are found. An alternative would be to find state invariants in order to rule out actions that would never succeed. State invariants can be automatically derived using a system like TIM [Fox and Long, 1998].

The general idea is that state invariants can be used to to identify actions that will never succeed. One type of state invariant are those facts that never change their truth value or can have their value changed. The `mystery` domain includes a predicate called `conn` which is used to create a link between two locations and the links can never be destroyed or created. Such predicates are invariants and can be identified as such. An action that relies on an invariant to be true, where it is false in the initial state, will never work and therefore it can be safely ignored.

#### 9.2.1.4 Summary

The action space, or amount of distraction in a domain, has been recognised as having an important impact on the efficiency of GENPLAN. There are some simple measures that can be taken to improve the situation, for example re-encoding the domain. Further improvements could be made to reduce the number of actions, by performing domain pre-processing.

### 9.2.2 Preparation Problem (Benches)

When first exploring the deficiencies of GENPLAN, the term "Preparation Problem" was coined to describe GENPLAN's inability to string together long sequences of actions to achieve a single goal. In particular when the fitness function gives no positive feedback towards achieving a goal. This was first noted in the `logistics` domain and continues to be a severe issue [Westerberg and Levine, 2000]. This problem was also noted in one of the first pieces of genetic planning work [Handley, 1994]. None of the default genetic operators, crossover or mutation, are defined to place plausible sequences of actions together without feedback.

The "Preparation Problem" is equivalent to benches, which form of various sizes when $ff_{gc}$ provides no information back to the algorithm. For instance, in the `maxones` domain its takes at most one action to achieve any goal. That is the fitness changes after the successful application of an action. For `briefcase` the issue gets slightly harder. It takes at least 3 actions to achieve a goal and the required sequence is `load`, `move`, `unload`. After the first two actions have been applied a candidate gets no fitness increase, it is only once the object has been unloaded in the correct location that a candidate get a fitness increase. The previous example makes a minimum bench size of 3.

#### 9.2.2.1 Benches - TTD_Conn

The issue of benches can be demonstrated using a variation of the `TTD` domain. With `TTD_Conn` there are explicit connections between cities, rather than assuming every city is connected to every other city. This is a relatively simple addition to make and the domain works semantically much like the `mystery` domain but without the resource constraints.

Using connections between cities it is possible to make long chains of cities. As Figure 9.3 shows the agent can be placed in ever increasing distances away from the

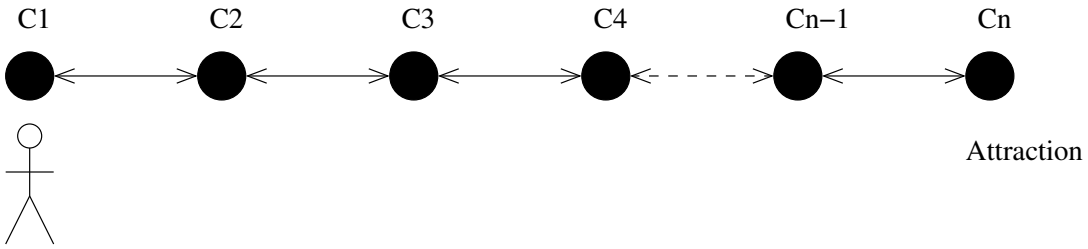



Figure 9.3: TTD_Conn Pathological Problem of size $N$

| Problem | Mean (failures) | std | median |
|---|---|---|---|
| pathological-6 | 1.52 (0) | 0.25 | 1.39 |
| pathological-7 | 3.69 (0) | 1.48 | 3.47 |
| pathological-8 | 18.06 (0) | 13.41 | 15.43 |
| pathological-9 | 242.41 (21) | 170.02 | 206.6 |
| pathological-10 | 489.14 (96) | 38.81 | 497.88 |

Table 9.1: The effect of increasing bench sizes on the efficiency of GENPLAN using TTD_Conn

attraction by adding additional cities. This enables the creation of problems to demonstrate the effect of a pathological case of ever increasing bench sizes for problems containing a single bench. If would be possible to simulate the effect of multiple benches by fanning out more points from the starting location.

Table 9.1 presents results of GENPLAN on a collection of TTD_Conn pathological problems. The problems are hand made and mirror Figure 9.3. The first problem, pathological-6, contains one bench of size 6. That is, GENPLAN must put together at least 6 actions (5 movement actions and one visit action) in order to achieve the goal of visiting the attraction. The bench size is increased by adding a city to the chain and placing the attraction and the agent at opposite ends of the chain. It is often the case that GENPLAN will use more actions than are necessary to achieve a goal. Table 9.1 shows that the efficiency of GENPLAN steadily decreases in the face of the steady increase in the size of the bench until GENPLAN can no longer reliably find a solution. These results demonstrate the effect benches can have on GENPLAN. The problem becomes worse as the complexity of domain rises in terms of size of the search space or the effect of local minima.

| Problem | Mean (failures) | std | median |
|---|---|---|---|
| pathological-1 | 1.34 (0) | 0.06 | 1.35 |
| pathological-2 | 1.35 (0) | 0.05 | 1.35 |
| pathological-3 | 1.34 (0) | 0.04 | 1.34 |
| pathological-4 | 1.37 (0) | 0.11 | 1.35 |
| pathological-5 | 2.07 (0) | 0.61 | 1.86 |
| pathological-6 | 1.39 (0) | 0.15 | 1.37 |
| pathological-7 | 12.58 (0) | 9.96 | 9.66 |
| pathological-8 | 46.11 (0) | 45.71 | 33.35 |
| pathological-9 | 259.34 (24) | 174.61 | 227.01 |

Table 9.2: The effect of increasing bench sizes on the efficiency of GENPLAN using `TTD_Conn` keeping the action space fixed

The problems used for Table 9.1 have increasing action spaces for each problem as the number of cities increases, from 49 to 121 actions. In order to rule out the effect of the action space, GENPLAN was run on another set of pathological problems. This time the number of cities is fixed to nine and the attraction is placed increasingly further away from the agent. The results are presented in Table 9.2. The number after pathological indicates the minimum number of actions to achieve the goal. This time the action space remains constant. GENPLAN very rapidly starts to fail once a bench size of 9 is reached.

Resolving this problem is an open issue for GENPLAN and so is the general problem of searching neutral or flat search spaces. However, in most planning problems there is an obvious structure that can be exploited. ADFs proved a means of creating short functions, or macros, that can be re-used in a candidate, and could encapsulate commonly occurring sequences of actions [Koza, 1992]. As has been highlighted a useful structure, that could be found automatically, is the `load`, `move`, `unload` sequence. This would reduce the bench size for some goals to 1 for the `briefcase` domain.

*Encapsulation* is another way to achieve modularisation. Encapsulation, in standard GP, is taking a fit candidate from the population and randomly selecting one of its subtrees. The subtree is shrunk to a single terminal and can be used as such. The new terminal can be added through mutation. Roberts used GP with encapsulation for target

detection from infra-red line-scan images [Roberts et al., 2001]. Using encapsulation he was able to evolve better solutions faster when compared with standard GP.

### 9.2.3 Misinformation Problem (Local Minima and Dead-ends)

The term "Misinformation Problem" was coined to describe GENPLAN's behaviour in the face of certain features of the search space. There are two main types of misinformation that $ff_{gc}$ heuristic reports back to GENPLAN. The first type is commonly come across in all areas of AI and is local minima. Local minima are bumps in the search space that are not optimal, or in this case not even solutions. In order to escape local minima the search algorithm has to search over less fit candidates in order to reach a solution. The second kind of misinformation is to provide positive feedback about a candidate when in fact the candidate results in a state from which a solution can never be found. These states are called *unrecognised* dead-ends, shortened to dead-ends for the rest of the chapter.

#### 9.2.3.1 Local Minima - Blocksworld

The effect of local minima of various sizes can be clearly demonstrated with pathological `blocksworld` problems. Figure 9.4 presents pathological problems using a variable height single stack. The local minima occur because the stack needs to be unstacked, destroying perceived achieved goals, in order to release `b1` so that the entire problem can be solved. The problems are constructed to contain ever larger local minima by increasing the number of blocks. The first problem has a local minima of size 3, or at least 3 actions in order to get to part of the search space where a real solution can be constructed. This is because the first problem has three out of four goals already achieved, giving a perceived fitness of 0.75. The problems increase with difficulty by increasing the size of the initial local minima, by increasing the number of blocks.

Referring to Table 9.3, the number after "pathological" indicates the size of the local minima. As the results show, the increasing local minima size, very rapidly creates problems for GENPLAN, making a 6 block problem almost unsolvable. Similarly with benches, it is also possible to create `blocksworld` problems with multiple variable sized local minima, creating even more difficulty, using multiple stacks.

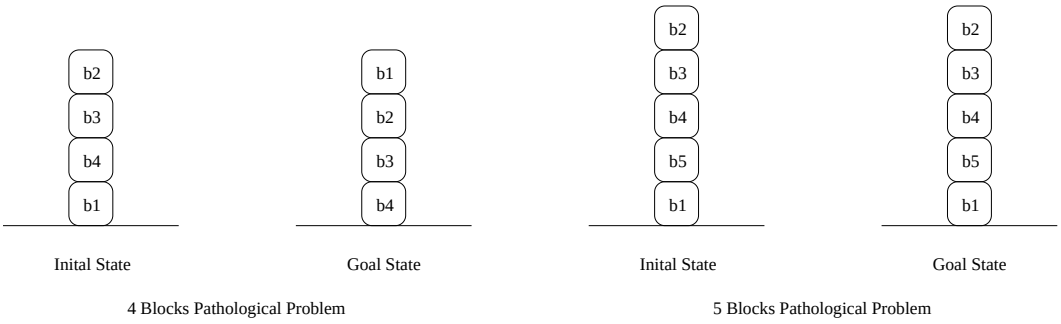



Figure 9.4: Two example `blocksworld` pathological local minima problems

| Problem | Mean (failures) | std | median |
|---|---|---|---|
| pathological-3 | 0.96 (0) | 0.03 | 0.95 |
| pathological-4 | 1.18 (0) | 0.31 | 0.99 |
| pathological-5 | 28.52 (0) | 23.24 | 22.66 |
| pathological-6 | 466.64 (85) | 90.56 | 497.56 |

Table 9.3: GENPLAN efficiency results on variable size local minima created using `blocksworld-3ops`

| Problem | Mean (failures) | std | Median |
|---|---|---|---|
| | Set 1 | | |
| 2-6-1-3-4-0 | 53.79 | 35.45 | 41.21 |
| 2-6-1-3-4-1 | 67.1 | 40.27 | 57.39 |
| 2-6-1-3-4-2 | 129.84 | 77.56 | 109.79 |
| | Set 2 | | |
| 2-7-1-3-5-0 | 83.75 | 56.93 | 75.59 |
| 2-7-1-3-5-1 | 142.42 | 82.64 | 129.42 |
| 2-7-1-3-5-2 | 220.61(4) | 121.97 | 189.29 |

Table 9.4: The effect on the efficiency of GENPLAN when varying the number of single action local minima for `satellite`

#### 9.2.3.2 Local Minima - Satellite

The `satellite` domain allows for one action local minima, by giving goal directions to any of the satellites in the problem. If the satellite is pointing in its goal direction but has image goals yet to be achieved then this goal must be disrupted if the image goal requires turning the satellite to a new direction. Two example problems are used from the previous chapter: 2-6-1-3-4, and 2-7-1-3-5. These problems originally used goal directions for both satellites. The problems are altered to include a single goal direction for a satellite and to having no goal directions for satellites. The efficiency results of GENPLAN on these problems are presented in Table 9.4. The final number in the problem name is an addition and shows the number of local minima in the problem. The efficiency results show that excluding the small local minima makes the main goals of taking images easier to achieve.

#### 9.2.3.3 Local Minima - DriverLog

The domain `driverlog` can have many one action local minima. This can be achieved by specifying goal locations for both drivers and trucks. If a truck is at its goal location, yet it has to be moved in order to move a package to its goal location, this creates a one action local minima. The problem taken from Chapter 8 is 3-3-2-2-2. The original specification involved three single action local minima. The local minima can be varied by changing the number of goal locations for drivers and trucks. The results for these

| Problem | Mean (failures) | std | Median |
|---|---|---|---|
| 3-3-2-2-2-0 | 34.5 (0) | 28.9 | 27.21 |
| 3-3-2-2-2-1 | 67.29 (0) | 65.04 | 39.74 |
| 3-3-2-2-2-2 | 59.76 (0) | 40.52 | 48.85 |
| 3-3-2-2-2-3 | 203.18 (12) | 144.28 | 147.62 |
| 3-3-2-2-2-4 | 265.68 (18) | 155.81 | 244.96 |

Table 9.5: The effect on the efficiency of GENPLAN when varying the number of single action local minima for `driverlog`

problems are presented in Table 9.5. The final number in the problem name gives the number of local minima. The results clearly show decreasing efficiency as additional single local minima are introduced, to what is essentially the same problem. The goal locations for packages are not altered.

#### 9.2.3.4 Unrecognised Dead-ends - Mystery

Referring back to Chapter 2, dead-ends come in four categories: undirected, harmless, recognised, unrecognised. The last of these, "unrecognised" is the most difficult to deal with. The only domain from the thesis set that can have unrecognised dead-ends in the search space is numeric `mystery`. In general, most classical planning domains fall into the first two categories of dead-end difficulty. Unrecognised dead-end states occur in numeric `mystery` when all the available fuel is used at a location so that the vehicle can no longer move from that location.

Table 9.6 illustrates the effect of continually decreasing the available fuel at each location, though the rest of the problem remains the same and the problems remain solvable. The problem 5-1-5-5-5 was chosen from the previous chapter. The problem has 5 locations, 1 vehicle, and there are 5 packages to deliver. The amount of fuel at each location was originally set to be equal to the number of locations in the domain. The amount of fuel starts at five and is decreased until there is one unit at each location (this version of the problem is still solvable). The results reveal that this search problem also creates difficulty for GENPLAN. The issue of dead-ends is related to the issue of resources in the next section.

| Problem | Mean (failures) | std | Median |
|---|---|---|---|
| 5-1-5-5-5 | 26.05 (0) | 6.34 | 25.1 |
| 5-1-5-4-5 | 28.25 (0) | 7.82 | 28.05 |
| 5-1-5-3-5 | 29.41 (0) | 6.64 | 28.05 |
| 5-1-5-2-5 | 35.49 (0) | 11.01 | 35.55 |
| 5-1-5-1-5 | 139.5 (13) | 158.09 | 64.24 |

Table 9.6: The effect of increasing the number of unrecognised dead-ends on the efficiency of GENPLAN using numeric `mystery`

#### 9.2.3.5 Summary

As has been shown, local minima in search can have an adverse effect on the efficiency of GENPLAN. A possible general strategy, that could apply to `blocksworld` like domains, would be include goal ordering information into the fitness function. Goal ordering mechanisms have been used positively with the FF planning system [Hoffmann and Nebel, 2001]. Also goal ordering has been helpful in a genetic planning context in [Yu et al., 2004].

### 9.2.4 Resources

Many different types of resources exist for planning problems [Long et al., 2000a]. The two of primary concern for this thesis are: restricted carrying capacity, and consumables. The first kind directly involves transportation domains where there is a limit to the amount that can be carried. The second type involves consumable resources like fuel, and the effect of which has already been examined in the previous section. A related resource issue is to include multiple carrying transportation objects, like multiple briefcases or satellites. In terms of difficulty, these additional objects will make problems more difficult, as they will increase the size of the action space.

#### 9.2.4.1 Gripper vs Briefcase

One interesting resource comparison that can be made from the previous chapter would be to compare the restricted carrying resource of `gripper` with the infinite resource of `briefcase`. New `briefcase` problems were made to mirror the problems used for

`gripper`: two locations, and varying the same number of objects from 1 to 14. The goal being to transport all object from one room to another. Figure 9.5 illustrates, in terms of efficiency, the effect of limited carrying capacity on problem difficulty. The figure shows that GENPLAN solves problems more efficiently when the carrying resource is infinite.

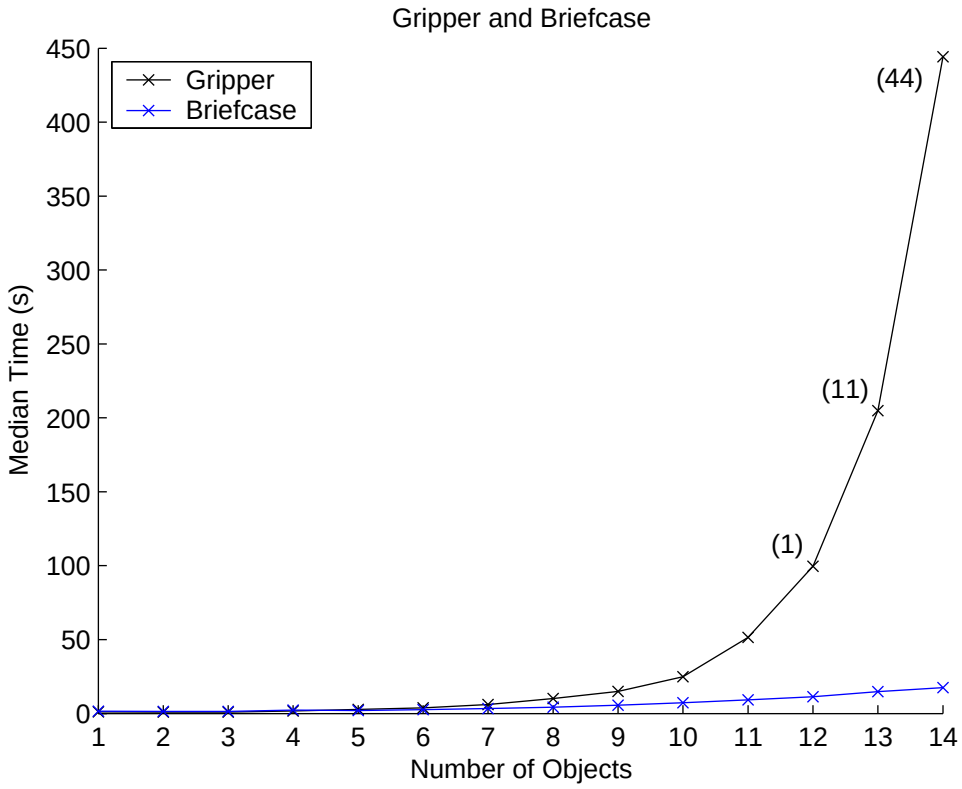



Figure 9.5: Comparing efficiency results for limited and unlimited carrying capacity using `gripper` and `briefcase`

## 9.3 Evolutionary Problems

The three main issues for evolutionary systems are diversity, bloat, and the crossover operator. The `maxones` domain is used to discuss the issue of diversity as it may be one cause of plan failure for this domain.

### 9.3.1 An investigation in to Action Diversity using Maxones

In the context of EAs, diversity is attempting to measure how different individuals in a population actually are either by examining their internal structure or by examining the solution they produce [Burke et al., 2002]. A lack of diversity is often cited as a key reason for failure in an EA. The diversity measure examined in this chapter is quite simplistic and is related to the make up of candidates.

The `maxones` domain is an interesting domain as it suffers from none of the aforementioned planning issues yet the performance of the algorithm does not scale gracefully with the increase in problem size when a restrictive maximum plan length is used. It was presented in Chapter 8 that maximum plan length has a significant role to play in determining whether a `maxones` problem will be solvable or not. This section approaches the problem from a different angle.

One way in which GENPLAN could fail outright for `maxones` is if an action becomes extinct in the population. For example, if the `switch(s20)` action is no longer present in the population then that switch can never be switched from off to on. Though the mutation operator, `VarAddMutation`, can protect against this, it is unlikely that it will add the correct missing action (depending on the size of the action space). This leads to a potential argument that diversity is being lost, so that a solution can no longer be found. It is possible for actions to become extinct during crossover when a child goes over the maximum plan length limit, then any action that should be copied into that child after the limit is ignored.

In order to investigate this claim I measured the diversity of actions at the end of a run. At the end of a run of GENPLAN the number of each action in the population can be counted. This will indicate the diversity of each switch action. This can be done over several runs recording the occurrence of each particular action at the end of each run. If these scores are then placed in a histogram, this will then indicate how many times there was a particular number of actions. For `maxones` I am interested if this is ever zero, or close to zero, for failed runs. Intuitively if there is less of a particular action then the problem will be harder to solve, as at least one of each action is required.

The histograms in Figure 9.6 report exactly what was described above. The problem chosen was the most difficult problem involving 120 switches. The counting was done for 100 successful runs and 100 failed runs, as it is interesting to see if different behaviour is exhibited between the two cases. The top two histograms present results

| Bin | Failed Value | Successful Runs Value |
|---|---|---|
| 0-50 | 97 | 40 |
| 51-100 | 132 | 61 |
| 101-150 | 141 | 91 |
| 151-200 | 206 | 130 |
| 201-250 | 298 | 197 |

Table 9.7: Specific frequencies for the first 5 bins from the all action histograms

when all actions present in the population at the end of a run are counted. The two lower histograms only consider the working actions at the end of the run.

For the failure cases I expect there to be more times where there are fewer actions for both cases (all actions and working actions). The histograms tentatively support this conclusion. Table 9.7 presents the bin scores for the first five bins and it appears it is more frequent that there are less actions of a particular type. This is detrimental as it means a plan that turns on all switches will less likely to be found. More investigation is required in to the population dynamics of GENPLAN.

### 9.3.2 Bloat

Bloat is an important EA issue, in particular to representations that can vary in size Bloat is the term given to the often very rapid growth of candidates during a run. Introns are also intimately involved with bloat and are loosely defined as sequences of code that do not achieve anything, either because they do not execute, form loops, or are not needed for the final solution. The reasons for the causes of bloat, its role in evolution, and the best way in which to counter bloat are still very much in contention [Langdon and Poli, 2002].

#### 9.3.2.1 Bloat during a run

The phenomena of bloat is present in GENPLAN. Figure 9.7 records mean candidate length of candidates between generation 1 to 50 over 100 runs. The problems used are from the parameterisation set and are the same four problems used to decide on mutation rates from Chapter 7. Each of the domains exhibit very rapid growth to the maximum possible, which is 200 actions. This shows that very sub-optimal results

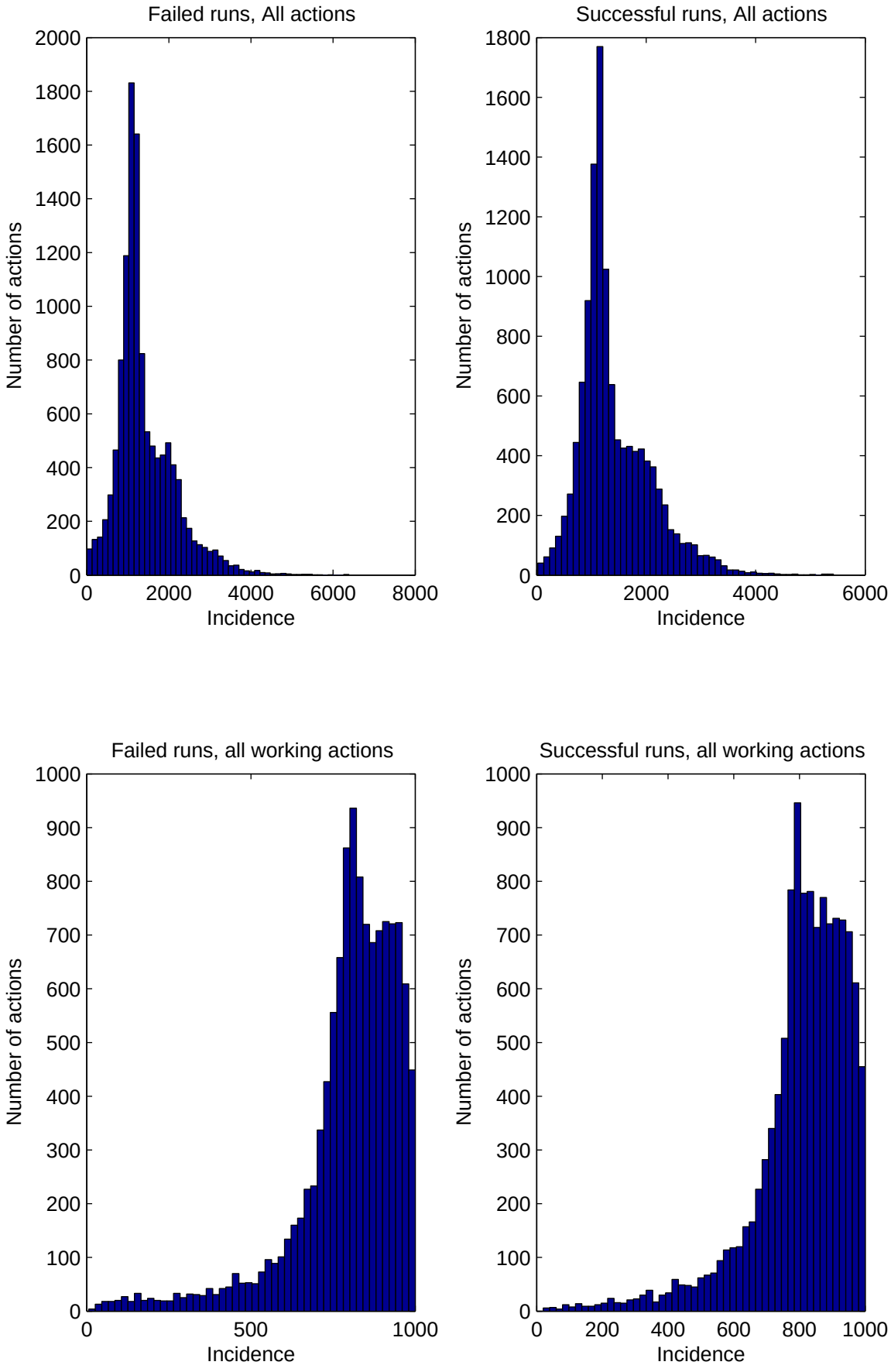



Figure 9.6: Histograms showing the frequency of different numbers of actions. The top two histograms are for all actions and the bottom two are for all working actions

can be produced. GENPLAN at least takes the precaution of stripping out non-working actions at the end of the run.

There have been many approaches to identify the causes of bloat and to reduce the amount of bloat in the GP literature [Langdon and Poli, 2002]. One interesting method is to use a "homologous" crossover. For this crossover method the two swapped parts are of approximate equal size [Langdon, 1999]. Another common way is to incorporate a length component into the fitness function. The best way to achieve this would be to use the length as a tie-break between equally correct plans. This is a type of non-parametric fitness function [Luke and Panait, 2002].

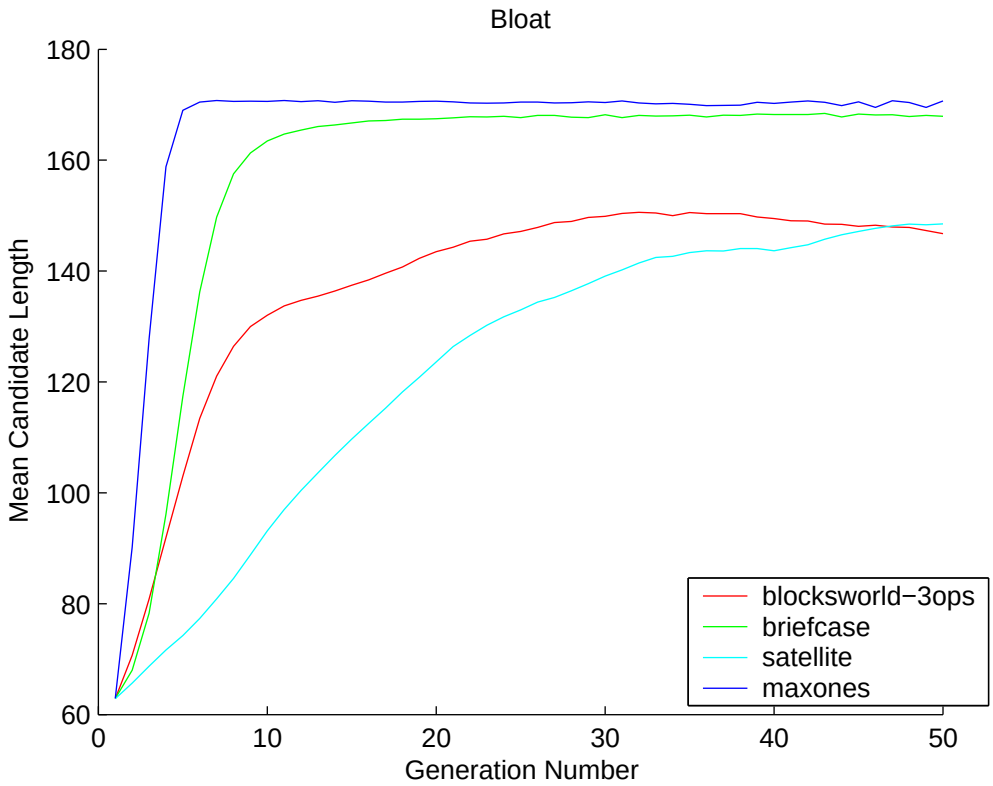



Figure 9.7: An example of bloat for the four parameterisation domains using problems: 8-003, 5-7-1, 1-9-1-2-4, 98 switches

### 9.3.3 Crossover

Crossover is an operator that has experienced much controversy in the EA community. Some argue that it is simply a macro-mutation and nothing more [Jones, 1995], and others argue that it is at the centre of how EAs work and work well [Goldberg, 2002].

One interesting test for the effectiveness of crossover used in the literature is comparing standard crossover with "Headless Chicken" crossover operators. One application of headless chicken crossover for tree-based GP is reported in [Angeline, 1997]. In this paper, a selected parent is crossed with a randomly generated tree. It was found that the headless chicken operator performed at least as well as regular crossover on a collection of problems. The motivation behind this experiment is that if a candidate is crossed with a randomly generated candidate, and the algorithm still performs as efficiently, then the crossover operator is not doing much more than random mutation.

I will repeat a similar experiment with GENPLAN. A 1-point crossover was created that behaves like standard crossover. Two parents are selected using tournament selection, but the second parent is then replaced with a randomly generated candidate of the same length as the original parent. Headless chicken crossover then behaves like standard crossover, the two parents are crossed and two children are produced and placed into the next generation. This has the effect of turning crossover into a macro-mutation operator.

In order to test the two different crossover operators a manipulation experiment is performed using the Mann-Whitney test. The test will seek to answer the question "Does standard crossover perform more efficiently than the equivalent headless chicken crossover operator." Keeping all other parameters and functionality the same, headless chicken crossover replaced standard crossover and GENPLAN was run over the standard parameterisation problems from Chapter 7. More explicitly the hypotheses for this test are:

- **Null Hypothesis:** The median CPU time to solve planning problems for GENPLAN using standard crossover is greater than or equal to the median CPU time to solve planning problems for GENPLAN using headless chicken crossover.
  **Alternative Hypothesis:** The median CPU time to solve planning problems for GENPLAN using standard crossover is less than the median CPU time to solve planning problems for GENPLAN using headless chicken crossover.

The two samples for this test are made up of median times to solve each problem using either headless chicken crossover or standard crossover. The standard crossover results are reused from Chapter 7 to make up the first sample and a new set of results are created using headless chicken crossover to make up the second sample. This results in a sample size of 20 for each crossover operator, one for each problem in the set. For this experiment $\alpha$ is set to 0.05.

| Problem | Mean (failures) | std | median | Problem | Mean (failures) | std | median |
|---|---|---|---|---|---|---|---|
| 86-001 | 9.89 (0) | 2.69 | 9.58 | 86-001 | 31.38 | 19.43 | 25.06 |
| 90-001 | 19.86 (0) | 5.79 | 19.7 | 90-001 | 81.03 | 46.8 | 69.39 |
| 94-001 | 32.95 (0) | 10.3 | 33.25 | 94-001 | 182.6 (2) | 111.27 | 155.65 |
| 98-001 | 49.89 (0) | 18.06 | 46.22 | 98-001 | 364.59 (47) | 154.76 | 457.86 |
| 102-001 | 88.24 (0) | 36.47 | 78.93 | 102-001 | 411.96 (60) | 126 | 498342 |
| 5-7-1 | 33.93 (0) | 55.86 | 22.35 | 5-7-1 | 387.68 (41) | 122.61 | 447.21 |
| 6-5-1 | 18.61 (0) | 5.49 | 17.68 | 6-5-1 | 170.01 (3) | 100.51 | 149.05 |
| 6-6-1 | 23.89 (0) | 22.11 | 17.7 | 6-6-1 | 227.51 (4) | 118.17 | 185.79 |
| 7-5-1 | 35.73 (0) | 32.15 | 24.31 | 7-5-1 | 381.53 (46) | 129.48 | 452.29 |
| 7-6-1 | 71.87 (0) | 74.34 | 38.88 | 7-6-1 | 454.22 (74) | 78.74 | 497.12 |
| 6-002 | 5.72 (0) | 1.71 | 5.49 | 6-002 | 19.47 (0) | 11.64 | 17.7 |
| 7-006 | 8.69 (0) | 2.77 | 8.51 | 7-006 | 40.47 (0) | 17.86 | 37.56 |
| 8-002 | 20.9 (0) | 15.4 | 16.51 | 8-002 | 92.19 (1) | 77.32 | 62.57 |
| 8-003 | 51.71 (0) | 55.84 | 25.66 | 8-003 | 232.72 (11) | 157.88 | 195.92 |
| 9-007 | 38.99 (0) | 24.51 | 30.88 | 9-007 | 219.97 (11) | 133.32 | 176.95 |
| 1-9-1-2-4 | 81.56 (0) | 44.65 | 78.61 | 1-9-1-2-4 | 284.96 (10) | 120.33 | 298.3 |
| 1-5-1-3-2 | 7.62 (0) | 5.97 | 6.09 | 1-5-1-3-2 | 19.52 (0) | 8.95 | 17.94 |
| 1-7-1-3-4 | 38.2 (0) | 23.07 | 33.66 | 1-7-1-3-4 | 213.81 (3) | 114.8 | 176.08 |
| 1-7-1-3-2 | 25.47 (0) | 17.64 | 21.09 | 1-7-1-3-2 | 60.08 (0) | 35.41 | 50.92 |
| 1-10-1-3-2 | 151.1 (1) | 104.03 | 122.98 | 1-10-1-3-2 | 222.93 (4) | 119.9 | 200.17 |

Table 9.8: CPU time results comparing standard crossover (left column) with headless chicken crossover (right column)

Table 9.8 presents efficiency results when each operator is used. For each of the domains the efficiency of GENPLAN is worse when using the headless chicken crossover. This conclusion is supported by the Mann-Whitney test which rejects the null hypothesis with a sum of ranks W of 48.5 and a p-value $<0.0001$. This is an encouraging result and confirms that standard crossover is performing a useful role during search. On the other hand due to the simplistic nature of the standard crossover operator, there should exist improved versions of crossover. For example, a more intelligent crossover could make more use of data generated during simulation, to select select crossover points more carefully.

## 9.4 Chapter Summary

This chapter presents the identified issues of the GENPLAN approach to domain independent classical planning. Most are general issues that any domain independent planner would have to over come: distraction, benches, local minima, and resources. Using an EA-based approach also brings additional issues: diversity, bloat, and the role of crossover. All of which direct areas in which to motivate improvements to alleviate these issues. All the issues are potentially severe, but for some of the issues potential solutions have been suggested. Implemented improvements to GENPLAN that address some of these issues are presented in the next chapter.

# Chapter 10

# Potential GENPLAN Improvements

This chapter describes the improvements made to some of the functional components of GENPLAN and the results of experiments using the improvements. Changes are made to the system in an attempt to improve the efficiency of GENPLAN but with plan quality remaining a factor. Each change to a component is described and observational experiments are performed to explore the effect on performance. Some improvements will also warrant a further manipulation experiment. Each improvement is related back to the issues presented in the previous chapter and how the improvement may alleviate some of the issues. The efficiency of GENPLAN is poor in comparison to modern fully independent planners and thus motivates any attempt at improving the efficiency of GENPLAN.

Four potential improvements are discussed in this chapter: two of which make use of simple search, one involving the fitness function, and the last involving domain engineering. A useful addition to GENPLAN was a general instantiation procedure which forms the basis of various kinds of simple search. The random search algorithm described in Chapter 2 is a good example of the type of search used. Simple search can be used in conjunction with the seeding component and with mutation operators. Two other improvements are discussed: an attempt to improve the fitness function for GENPLAN by incorporating a working action component into the fitness calculation and an idea is discussed for engineering domains to make them more suitable for GENPLAN.

## 10.1 Experimental Methodology

This chapter consists of a series of experiments in order to determine the effective use and affect on performance of the various improvements. For each improvement

an initial experiment is performed to evaluate the improvement and to parameterise the improvement if necessary. A manipulation experiment can be used to confirm the effect the improvement has on both efficiency and plan quality. The manipulation experiment requires the specification of the problem sets used, a statement of the null hypothesis and alternative hypothesis, and the samples being tested.

### 10.1.1 Choose Goals

The purpose of these experiments are two-fold: firstly to examine whether an improvement performs generally well and secondly to more closely examine improved performance in terms of efficiency and any trade-off with plan quality. This will be achieved by running GENPLAN with the improvement in question and comparing its behaviour with the original version of GENPLAN over a range of domains and problems.

### 10.1.2 Choose Metrics

The standard efficiency and quality metrics will be adopted for this chapter: median CPU time, and median plan length. The justifications and the means by which the metrics were recorded was presented in Chapter 6.

### 10.1.3 Choose Problems

The first subsection describes the choice of domains. The subsequent subsection describes the choice of the problems to those domains.

#### 10.1.3.1 Choosing Domains

The standard parameterisation domains are re-used for this chapter as they represent a limited but broad range of difficulty, see Table 10.1 for a summary of the domains. The domains contain most of search issues of interest: distraction, local minima and benches.

#### 10.1.3.2 Choosing Problems to the Domains

As with the domains, the standard parameterisation problems are re-used:

- `maxones`: These problems involve 86, 90, 94, 98, and 102 switches.
- `briefcase`: 5-7-1, 6-5-1, 6-6-1, 7-5-1, 7-6-1

| Domain | Dead-ends | Number of Actions | Max. Local Minima | Min. Bench Size |
|---|---|---|---|---|
| Maxones | harmless | 1 | 0 | 1 |
| Briefcase | undirected | 3 | 1 | 3 |
| Blocksworld-3ops | undirected | 3 | $< c*$ | $< c$ |
| **Satellite** | harmless | 5 | $1*$ | 3 |

Table 10.1: The selected domains for this chapter

- `blocksworld-3ops`: 6-002, 7-006, 8-002, 8-003, 9-002
- `satellite`: 1-9-1-2-4, 1-5-1-3-2, 1-7-1-3-4, 1-7-1-3-2, 1-10-1-3-2

Some experiments require varying two parameters simultaneously. For these experiments it was necessary to report results from a single problem. The chosen problems are identical to those used in Chapter 7 and they are:

- `maxones`: 98
- `briefcase`: 5-7-1
- `blocksworld-3ops`: 8-003
- `satellite`: 1-9-1-2-4

For those improvements that warrant a further manipulation experiment the descriptions and justification for further problem sets are given with the experiment.

### 10.1.4 Design of Experiments

The observational experiments are done in an identical fashion to the parameterisation experiments in Chapter 7. There is one caveat in that any identified improvements are not used in the other experiments within the chapter, that is the improvement is always a one step change from the default version of GENPLAN. The experiment in Section 10.2.3 is the only exception incorporating two improvements at once. The manipulation experiments are performed as specified in Chapter 6.

### 10.1.5 Running the Experiment

The experiments were performed in the standard way using 100 runs per problem and the standard parameters from Chapter 7. Any change in the parameterisation of GENPLAN is described.

### 10.1.6 Presenting the Results

The results in this chapter are an initial study into the behaviour of the implemented improvements. For each improvement I expect the efficiency of planning to improve and for there to be a trade-off with plan quality. The standard plots and tables are used for this chapter.

## 10.2 The Improvement Experiments

The experiments related to improvements are presented in no particular order though the experiments involving seeding and mutation use the same underlying simple search method and are presented together. The subsequent subsections describe the improvement and report results of incorporating the improvement into GENPLAN.

### 10.2.1 Improved Seeding Using Simple Search

The first potential improvement that will be discussed is an attempt to improve the population initialisation algorithm or seeding step. The motivation behind this improvement is that random selection of actions from the entire set of all possible actions for initial candidates will not place plausible sequences of actions together, thus contributing to the problem of distraction and benches. In terms of the bench issue, sequences of actions selected at random are not likely to produce sequences that achieve goals. In terms of distraction, it may be better to focus GENPLAN early on with more plausible sequences of actions, but it may also be necessary to include some random actions as to not make search too constrained.

One way to improve the quality of the initial population would be to initialise plans using a random walk in the state space. Starting from the initial state $S_0$, all actions that apply to that state are found, the set $A_s$, and an action, $a$, is chosen at random from that set and added to the initial plan. The state is updated by applying $a$. From the new state the process continues until the initial plan length is achieved. This will

place sequences of actions in the population that go together in the initial candidates. Improving the quality of the initial population has been found to be beneficial before [Westerberg and Levine, 2001a]. What follows in this thesis is similar to the paper but is over a more limited set of seeding strategies, but experiments are performed over a greater range of domains and problems.

One example of a positive result from improving the seeding algorithm is reported in [Langdon and Nordin, 2000]. In Langdon's work the author starts by seeding the population with perfect individuals and then asks the GP algorithm to find a general solution. The author compared this method with a random seeding strategy. The population based on perfect individuals was able to generalise better than the random method. This is an indication of the fact that there are benefits to improving the initial population seeding method rather than resorting to a simple random method.

Two seeding strategies are examined in this chapter called Strategy 0 and Strategy 1. Strategy 0 is the default random initialisation of populations and was presented in Chapter 5. Strategy 1 is a new strategy that incorporates simple search.

#### 10.2.1.1 Strategy 1: Random Walk plus Random Actions

This is the seeding strategy that will be investigated. During candidate initialisation the strategy either selects an action at random from the set of all possible actions or the strategy selects an action which applies in the current state. The probability of selecting either is set by a parameter $P_r$ called "percent random." A value of 0.1 means that there is 10% probability of selecting an action at random and a 90% probability of selecting an action from simple search. This additional parameter is included as populations created entirely random search may produce initial populations which are not diverse enough. Algorithm 4 summarises the strategy.

#### 10.2.1.2 Parameterisation Discussion

For the parameterisation experiment there is only one variable to vary, the ratio of actions selected at random and from simple search, $P_r$. The 100% value is equivalent to Strategy 0 and 0% is equivalent to seeding plans by simple search only. As the selected range is fairly broad it was not necessary to run a pre-parameterisation experiment. All other parameters are at their default values.

- $P_r$ **values** 0%, 10%, 20%, 30%, 40%, 50%, 60%, 70%, 80%, 90%, 100%

**Algorithm 4** Creates a random population of plans, using a combination of random walking and random actions

```
1: for 0 to Population Size do
2:    initialise the current state to S_0
3:    for 0 to Initial Length do
4:       Generate a random number r between 0 and 1
5:       if r < P_r then
6:          Select a random action
7:          Update the current state with the selected action if possible
8:       else
9:          Find all actions that apply in the current state, A_s
10:         Select an action from A_s at random
11:         Update the current state
12:      end if
13:   end for
14: end for
```

#### 10.2.1.3 Initial Results

The intuition for this improvement is that Strategy 1 should help in domains with larger bench sizes as it is more likely Strategy 1 will place plausible sequences of actions together in the initial individuals. This in turn should have a benefit on efficiency. As a downside, because Strategy 1 is placing more working actions in the initial population, I expect the plan quality to degrade as $P_r$ decreases. An additional downside of Strategy 1 is that it requires more computation to initialise the population as more effort is expended performing the search, compared with Strategy 0.

Figure 10.1 plots median CPU time against different settings of $P_r$.

- **Maxones:** This domain shows a clear upwards trend for time as $P_r$ decreases. This is probably due to the overhead incurred by performing simple search during the population initialisation phase.
- **Briefcase:** This domain shows a sharp downwards trend with even a slight decrease in $P_r$. The improvement stops after about 0.6.
- **Blocksworld-3ops:** These results are very similar to the `briefcase` results, with the best value around the 0.7 mark. For some values of $P_r$ a correct plan can be occasionally guessed at the outset but only for the easier problems.

- **Satellite:** The best values lie in the range of 0.7 to 0.0. Occasionally correct plans are guessed but only for some of the problems and settings.

- **Summary:** The domains show two results. For three of the domains Strategy 1 can improve the efficiency of planning. So much so in some limited cases the main algorithm does not come into play. The best value seems to support a mixture of search and random actions.

Figure 10.2 plots median plan length against different settings of $P_r$.

- **Briefcase:** This domain presents a linear growth in plan length as $P_r$ decreases.

- **Blocksworld-3ops:** This domain presents a linear growth in plan length as $P_r$ decreases. The increase in growth is less severe than that of `briefcase`.

- **Satellite:** The plot presents a mixture of growth rates for the different problems.

- **Summary:** As expected Strategy 1 had a detrimental affect on overall plan quality. The amount of search used to create individuals should be kept to a minimum, in terms of plan quality.

Certainly for the `blocksworld-3ops` domain and the `briefcase` domain there is considerable improvement in efficiency to the detriment of plan quality. The `satellite` domain presents its own problems as a guessing strategy nullifies the need for any genetic search, though this is not true for all problems and all values of $P_r$. It would be interesting to investigate how much further seeding can take GENPLAN in terms of problem difficulty for the domains.

#### 10.2.1.4 Further Results

The purpose of this section is to answer the question "Does Strategy 1 improve the efficiency of planning and decrease the quality of returned plans?" On the basis of the parameterisation experiments it is speculated that the efficiency of GENPLAN has in general been improved when using Strategy 1 for some values of $P_r$. In addition it is speculated that the quality of resulting plans has decreased when using Strategy 1. Using the parameterisation results as a guide, a value of $P_r = 0.6$ was chosen for the two manipulation experiments designed to answer the two questions. The experiments use the standard parameters except when it is necessary to change the seeding component.

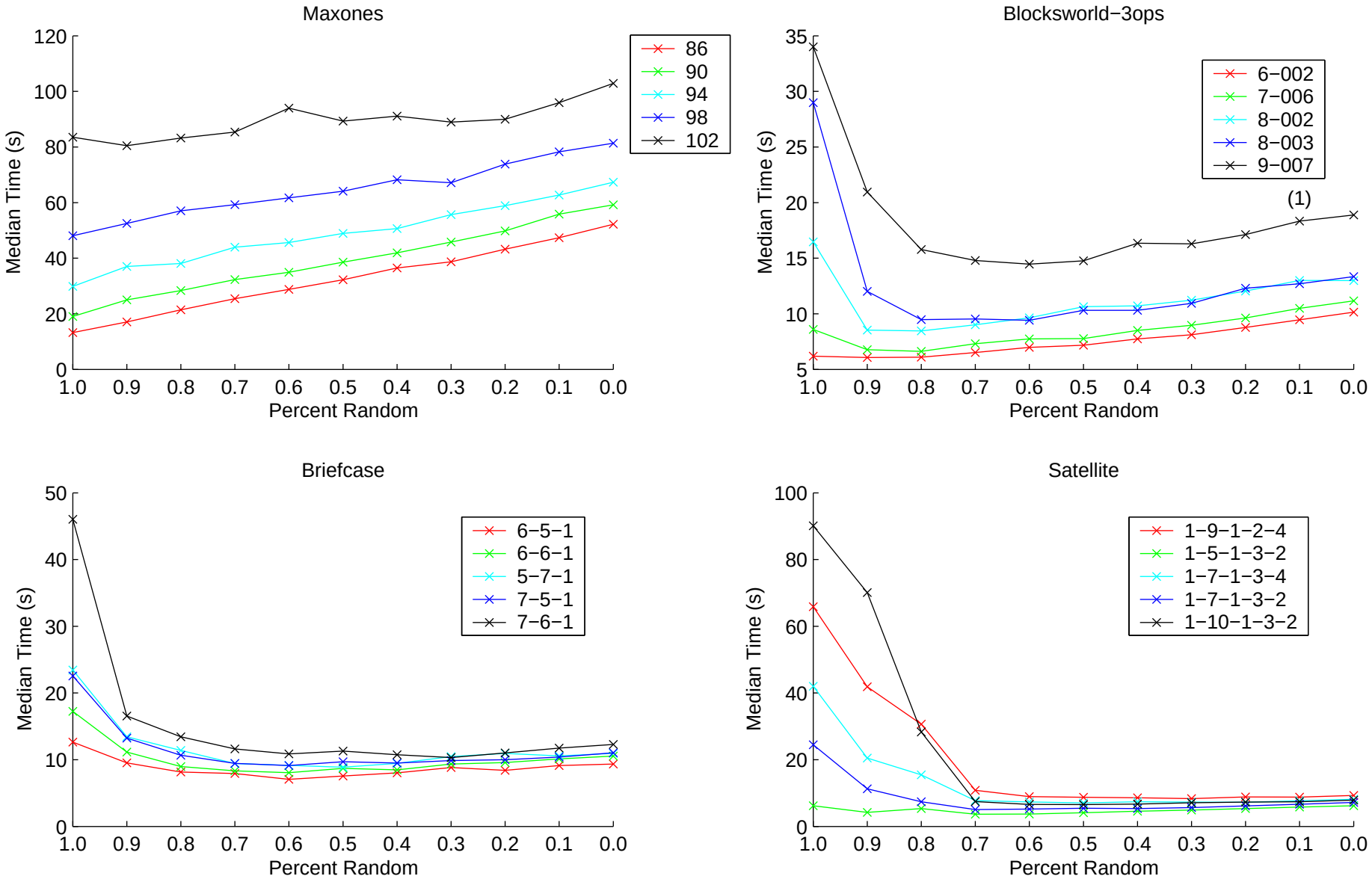



Figure 10.1: The effect of varying $P_r$ on the efficiency of GENPLAN

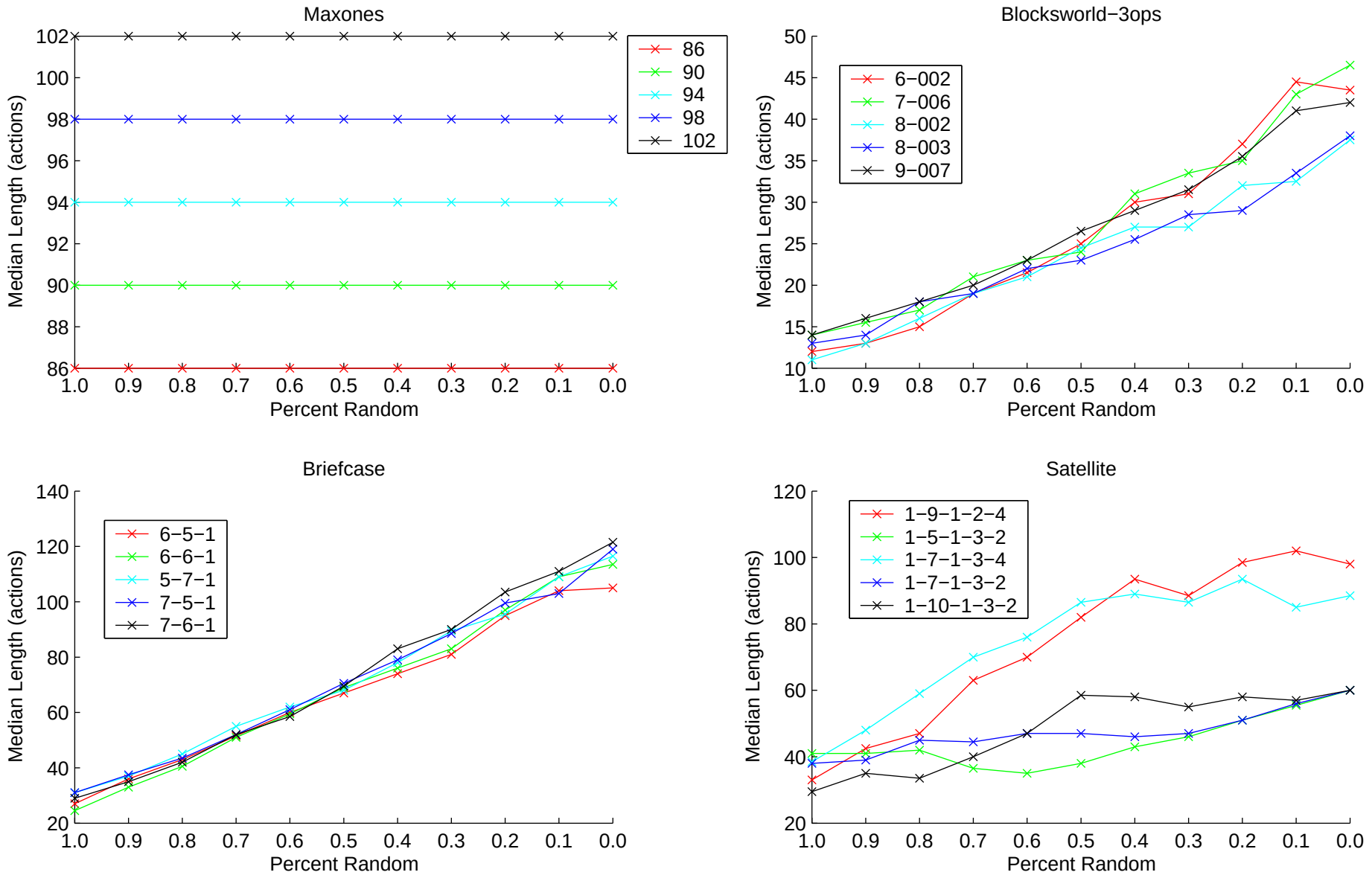



Figure 10.2: The effect of varying $P_r$ on the plan quality of returned plans

The manipulation experiment is run almost identically to the parameterisation experiment but including the changes stated in Section 10.1. The selected problems are all from Chapter 8. For the `blocksworld-3ops` problems the 8 block problems were used as this set includes both difficult and easy problems. For `briefcase` the hardest set of problems involving 9 locations were used. For `maxones` the five hardest problems were used, problems involving 100, 105, 110, 115, and 120 switches. For `satellite` the 2-*-1-3-* problems were used. The first star is number of directions (3-7) and second star is the number of goals (1-5). For each problem GENPLAN was run 100 times for both settings.

The manipulation experiment will incorporate a Mann-Whitney test as set out in Chapter 6. For this manipulation experiment $\alpha$ is set to 0.05. The following hypotheses are tested:

- **Null Hypothesis:** The median CPU time to solve planning problems for GENPLAN using Strategy 0 is less than or equal to the median CPU time to solve planning problems for GENPLAN using Strategy 1.
  **Alternative Hypothesis:** The median CPU time to solve planning problems for GENPLAN using Strategy 0 is greater than the median CPU time to solve planning problems for GENPLAN using Strategy 1.

- **Null Hypothesis:** The median plan length of resulting plans found with GENPLAN using Strategy 0 is greater than or equal to the median plan length of resulting plans found with GENPLAN using Strategy 1.
  **Alternative Hypothesis:** The median plan length of resulting plans found with GENPLAN using Strategy 0 is less than the median plan length of resulting plans found with GENPLAN using Strategy 1.

A sample of median times and a sample of median lengths will be generated from GENPLAN using Strategy 1 and the previous results using the default parameterisation from Chapter 8 will make up the second set of samples. The sample size for the manipulation experiment involving efficiency is 26 as that is the total number of problems used. The sample size for plan quality experiment is 21 as the results from `maxones` are not used.

Figure 10.3 plots median CPU time for both Strategy 0 and Strategy 1 using $P_r$ = 0.6 on the selected problems and domains. The lines in black are the original results from Chapter 8 and the blue lines are the new results with Strategy 1.

- **Maxones:** For this domain Strategy 1 has produced very similar results to Strategy 0. The parameterisation results for Strategy 1 suggest that it should be performing slightly worse. This could be due to two reasons: firstly due to the search overhead of Strategy 1 superseding any benefit of the additional search, or secondly that the additional search of Strategy 1 incurs no benefit for this domain.

- **Briefcase:** The efficiency of planning for this domain has improved. This could be because Strategy 1 is adding useful structures into the population that help GENPLAN overcome the issue of benches.

- **Blocksworld-3ops:** The efficiency has improved for some of the problems. Those problems that were hard using Strategy 0 remain hard when Strategy 1 is used, possibly due to the fact that this is the only domain to include variable size local minima. Also some additional failures have been recorded for the easier problems, possibly due to a lack of diversity.

- **Satellite:** The efficiency and reliability of planning for the selected problems has improved. As in the `briefcase` domain, Strategy 1 may be adding useful sequences of actions to alleviate the issue of benches.

- **Summary:** As predicted by the initial experiment the problems are generally being solved more efficiently using Strategy 1. This is confirmed by the Mann-Whitney test which rejects the null hypothesis with a p-value of 0.0019 and a sum of ranks $W$ of 494.

Figure 10.4 plots median plan length for both Strategy 0 and Strategy 1 using a $P_r$ = 0.6 on the selected problems and domains. The lines in black are the original results from Chapter 8 and the blue lines are the new results with Strategy 1.

- **Briefcase:** Plan lengths have nearly tripled when Strategy 1 is used as compared with Strategy 0.

- **Blocksworld-3ops:** Plan lengths have doubled when Strategy 1 is used as compared with Strategy 0.

- **Satellite:** Like the `briefcase` domain median plan lengths have increased dramatically when Strategy 1 is used to seed the initial populations.

- **Summary:** All the relevant domains have experienced a decrease in plan quality as predicted by the initial results. This can be expected as Strategy 1 adds many more working actions into the initial population. This is confirmed by the Mann-Whitney test as the null hypothesis is rejected with a p-value $<0.0001$ and a sum of ranks $W$ of 60.5.

The further results using improved seeding via simple search are quite encouraging. The efficiency of GENPLAN has improved and has been confirmed by experiment. The downside of the improved seeding is a loss of plan quality. These experiments are only initial results with this type of seeding, there are potentially many more types of search that can be used in place of simple search.

### 10.2.2 Improved Mutation Using Simple Search

The second improvement, which also makes use of simple search, is a new mutation operator. This mutation operator has similar motivations to the improved seeding component discussed in the previous section. The default `VarAddMutation` is unlikely to add useful actions to a plan at a random point by chance. The main purpose of this simple mutation operator is to add actions into the population and was shown to be of some benefit in Chapter 7.

A mutation operator based on simple search will instead add working actions at a specific point in the plan using simple search. This type of mutation may help overcome the issues of diversity, benches, and distraction. The mutation will help with diversity as the genetic operator will add actions that are needed at a specific point in the plan. The mutation may also help overcome benches if it achieves a goal and by also potentially placing useful sequences of actions in candidates. It will also help with distraction, as it will give GENPLAN potentially more useful actions to work with.

#### 10.2.2.1 Forward Chaining Mutation

A mutation operator called "forward chaining mutation" (FCM) was implemented incorporating simple search. The mutation works by adding $0-n$ actions to a plan at a random point $p$ in the plan. The plan is simulated up to the point $p$ then simple search is performed to a depth between $0-n$. The path through the search tree is recorded and the actions are added to the plan at point $p$. As with seeding, this kind of search is customisable and any kind of search could be used as a basis.

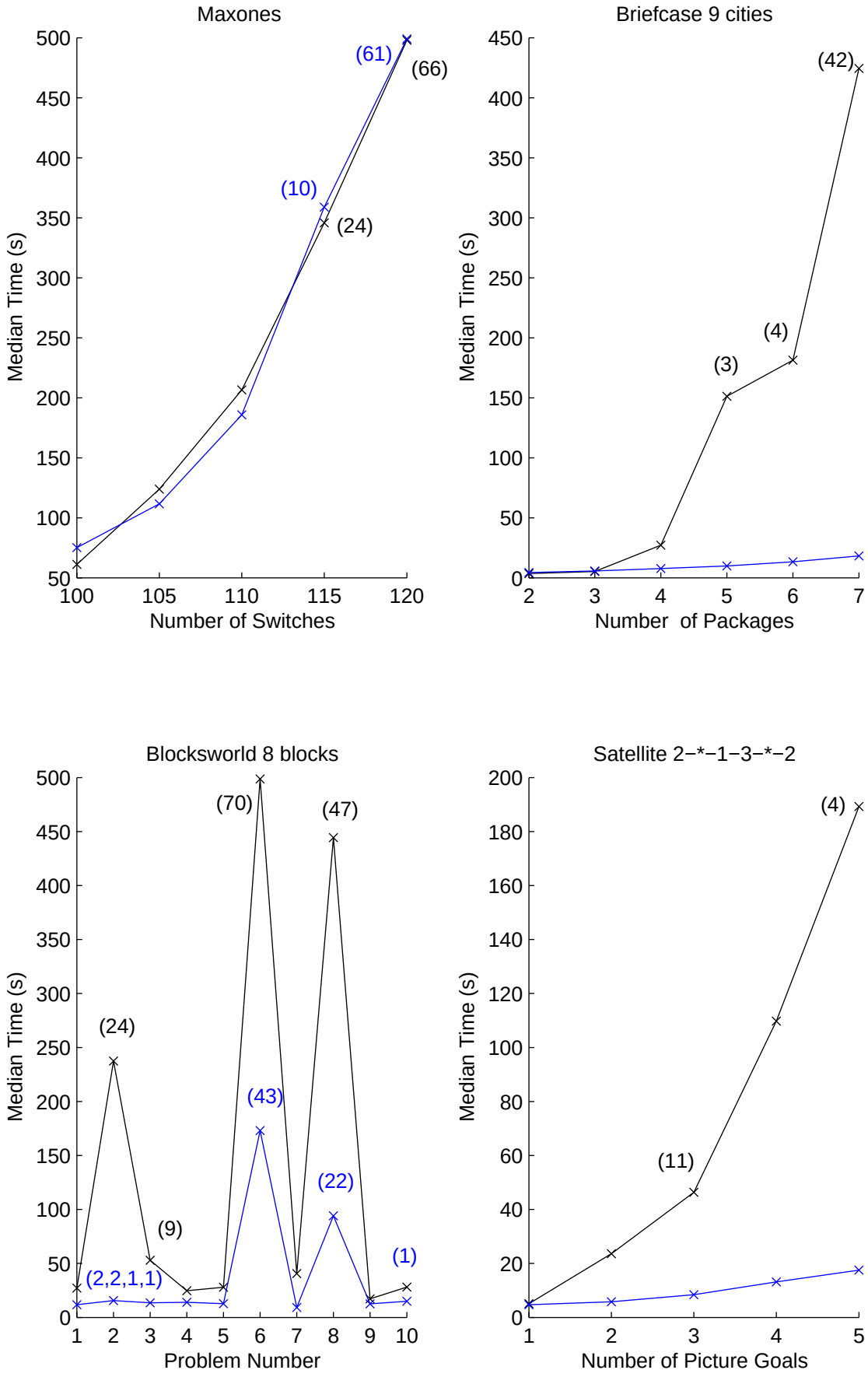



Figure 10.3: The effect of Strategy 1, $P_r = 0.6$, on efficiency when compared with Strategy 0

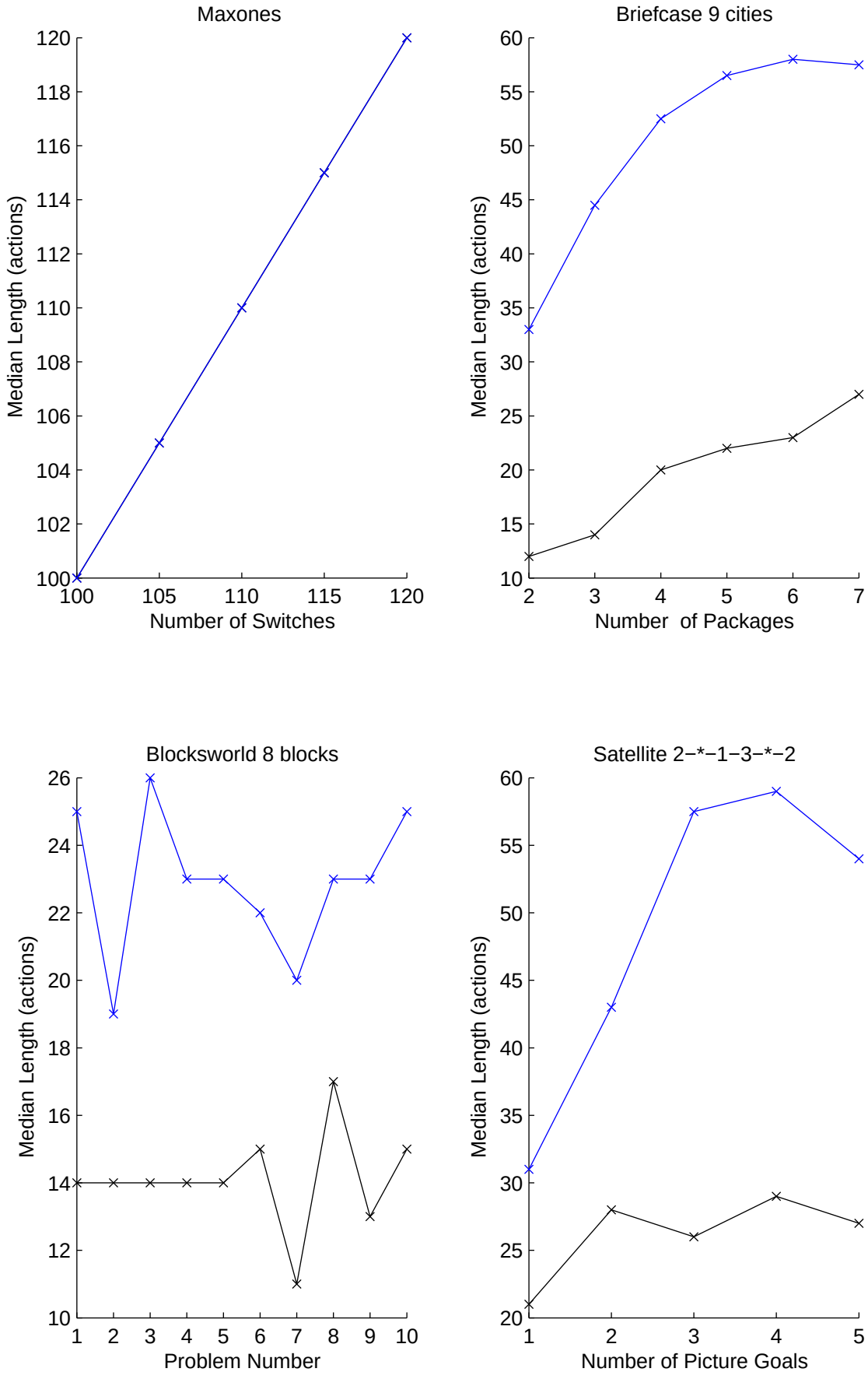



Figure 10.4: The effect of Strategy 1, $P_r = 0.6$, on plan quality when compared with Strategy 0

#### 10.2.2.2 Parameterisation Discussion

The parameterisation experiment for FCM was performed in the same manner as for the `VarAddMutation` experiment in Chapter 7. For this experiment, the `VarAddMutation` is switched off and replaced with the new FCM mutation. There are two parameters to vary for this experiment: the frequency of the mutation and the amount of actions that are added. The reduced parameterisation problem set is used for similar reasons as the `VarAddMutation` parameterisation experiment in Chapter 7. The values used for the experiment are:

- **Frequency:** 5%, 10%, 15%, 20%, 25%
- **Amount:** 0-5, 0-10, 0-15, 0-20, 0-25 (actions)

#### 10.2.2.3 Initial Results

The intuition for this improvement is that the efficiency of GENPLAN will improve as both the amount and frequency of mutation is increased. FCM should benefit search as the operator adds more working actions into the population, in particular in potentially usefully plan fragment sizes. Again as the mutation operator is adding working actions to candidates within the population I expect the quality of returned plans to decrease.

Table 10.2 presents median time results for the four selected problems. The best values are highlighted in bold.

- **Maxones:** The best values for this domain are derived when the amount is set to 0-25 actions. Increasing the frequency of the mutation can improve efficiency for some mutation amounts.
- **Briefcase:** The best values appear in the 0-25 column with increased frequency having a small beneficial impact on efficiency for each amount of mutation.
- **Blocksworld-3ops:** For this domain the best values appear to be located around 15% and 0-20 actions.
- **Satellite:** For this domain more efficient values are located around 15% and 0-15 actions.
- **Summary:** Improved efficiency results appear to have been found using the new mutation operator. Better efficiency results came from using a frequency around 15% with an amount of mutation around 0-20 actions.

Table 10.3 presents median plan length results for the four selected problems, again the the best values are highlighted in bold.

- **Briefcase:** The best quality plans are returned when the mutation is adding the lowest amount (0-5 actions). Increasing the amount of mutation has quite a detrimental affect on plan quality.
- **Blocksworld-3ops:** This domain shows the least effects for increasing amounts of mutation. The best values are again found in the 0-5 mutation amount column.
- **Satellite:** The plan quality for this domain suffers quite dramatically for increasing amounts of mutation, like the `briefcase` domain. Using the amount, 0-5 actions, produced the shortest plans.
- **Summary:** For all three domains it is better to use lower amounts of mutation when plan quality is considered. This is as expected as the mutation works to insert working actions into each candidate.

The efficiency results for this operator are quite good, having a beneficial effect for all four domains. However until additional improvements are added to GENPLAN in order to alleviate plan quality issues any selected mutation frequency and amount should be kept reasonably low. The pre-parameterisation results are encouraging, and a manipulation experiment is performed to investigate FCM as appears to be a promising operator.

| *Freq* | *Amount* | | | | | *Freq* | *Amount* | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | 0-5 | 0-10 | 0-15 | 0-20 | 0-25 | | 0-5 | 0-10 | 0-15 | 0-20 | 0-25 |
| | Maxones | | | | | | Blocksworld-3ops | | | | |
| 5% | 43.58 | 31.92 | 16.81 | 11.11 | **8.46** | 5% | 24.52 (1) | 20.85 | 19.48 | 17.85 | 16.52 |
| 10% | 44.65 | 27.57 | 13.69 | 10.83 | **9.22** | 10% | 19.13 (1) | 13.79 | **13.04** | **13.52** | 15.24 |
| 15% | 44.68 | 24.69 | 12.73 | **10.8** | **8.55** | 15% | 18.33 | 13.9 | 14.06 | **12.92** | **12.85** |
| 20% | 46.14 | 23.65 | 13.61 | 11.71 | **9.59** | 20% | 17.12 | 14.35 | **13.43** | 13.76 | 13.68 |
| 25% | 46.62 | 25.77 | 14.83 | 13.16 | **10.82** | 25% | 15.06 | 14.07 | 14.42 | **13.46** | 14.28 |
| | Briefcase | | | | | | Satellite | | | | |
| 5% | 16.67 | 14.17 | 11.93 | 11.11 | **9.95** | 5% | 34.66 | 17.94 | 12.64 | 10.76 | 9.68 |
| 10% | 17.67 | 12.93 | 11.18 | 10.26 | **9.95** | 10% | 25.16 | 14.07 | 11.06 | **7.4** | **9.21** |
| 15% | 16.78 | 12.38 | 10.47 | **9.68** | **9.99** | 15% | 19.65 | 12.66 | 10.41 | **9.17** | 12.9 |
| 20% | 15.67 | 12.58 | 10.93 | 10.35 | **9.91** | 20% | 18.6 | 11.29 | **8.16** | 9.37 | **9.05** |
| 25% | 15.52 | 12.4 | 11.2 | 10.88 | 10.61 | 25% | 17.53 | 12.23 | 10.1 | 9.32 | 15.63 |

Table 10.2: Median time results for FCM on the 4 parameterisation problems

| *Freq* | *Amount* | | | | | *Freq* | *Amount* | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | 0-5 | 0-10 | 0-15 | 0-20 | 0-25 | | 0-5 | 0-10 | 0-15 | 0-20 | 0-25 |
| | Maxones | | | | | | Blocksworld-3ops | | | | |
| 5% | 98 | 98 | 98 | 98 | 98 | 5% | **13** | 15 | 16 | 19 | 18 |
| 10% | 98 | 98 | 98 | 98 | 98 | 10% | **14** | 16 | 17 | 19 | 19.5 |
| 15% | 98 | 98 | 98 | 98 | 98 | 15% | **14** | 16 | 19 | 21 | 21 |
| 20% | 98 | 98 | 98 | 98 | 98 | 20% | **14** | 16.5 | 18 | 21 | 24 |
| 25% | 98 | 98 | 98 | 98 | 98 | 25% | 15 | 16 | 20 | 22 | 23 |
| | Briefcase | | | | | | Satellite | | | | |
| 5% | **32** | 38 | 44 | 50.5 | 58.5 | 5% | **42** | 57.5 | 66 | 74 | 78 |
| 10% | **33.5** | 42 | 47 | 56 | 64 | 10% | **48** | 61 | 71.5 | 77 | 83 |
| 15% | **35** | 45 | 51 | 59 | 70 | 15% | **49** | 64 | 70 | 79.5 | 84.5 |
| 20% | **35** | 44 | 54 | 62.5 | 67 | 20% | **49.5** | 59 | 71.5 | 78.5 | 85 |
| 25% | 36 | 45 | 57 | 64 | 69.5 | 25% | **50** | 64 | 74 | 79.5 | 89 |

Table 10.3: Median plan length results for FCM on the 4 parameterisation problems

#### 10.2.2.4 Further Results for Forward Chaining Mutation

The purpose of this section is to answer the question "Does FCM improve the efficiency of planning and decrease the quality of returned plans?" On the basis of the parameterisation experiments it is speculated that the efficiency of GENPLAN has in general improved when using FCM. In addition, it is speculated that the quality of resulting plans has decreased when using FCM. Using the parameterisation results as a guide the chosen parameter setting for FCM is 10%, 0-10 actions, as it is a good compromise of speed and quality.

The manipulation experiment is run almost identically to the parameterisation experiment but includes the changes stated in Section 10.1. The same problems used for the previous improved seeding experiment are re-used here. For each problem GENPLAN was run 100 times for both settings. The experiments use the standard parameters except when it is necessary to swap the mutation component.

The manipulation experiment will incorporate a Mann-Whitney test as set out in Chapter 6. For both experiments $\alpha$ is set to 0.05. The following hypotheses will be tested:

- **Null Hypothesis:** The median CPU time to solve planning problems for GENPLAN using `VarAddMutation` is less than or equal to the median CPU time to solve planning problems for GENPLAN using FCM.
  **Alternative Hypothesis:** The median CPU time to solve planning problems for GENPLAN using `VarAddMutation` is greater than the median CPU time to solve planning problems for GENPLAN using FCM.

- **Null Hypothesis:** The median plan length of resulting plans found with GENPLAN using `VarAddMutation` is greater than or equal to the median plan length of resulting plans found with GENPLAN using FCM.
  **Alternative Hypothesis:** The median plan length of resulting plans found with GENPLAN using `VarAddMutation` is less than the median plan length of resulting plans found with GENPLAN using FCM.

A sample of median times and a sample of median lengths will be generated with FCM mutation and the previous results using `VarAddMutation` from Chapter 8 will make up the second sample. The sample size for the manipulation experiment involving efficiency is 26 as that is the total number of problems used. The sample size for plan quality is 21 as the results from `maxones` are not used.

Figure 10.5 plots median CPU time for both `VarAddMutation` and FCM on the selected problems and domains. The lines in black are the original results from Chapter 8 and the blue lines are the new results with FCM.

- **Maxones:** The FCM operator allows problems to be solved more efficiently.
- **Briefcase:** For this domain, problems are being solved more efficiently and with less failures when the FCM operator is used.
- **Blocksworld-3ops:** Only some of the problems are being solved more efficiently for this domain. The problems which were hard in the first instance, for example problem 6, remain hard, as this could be due to some other search issue that needs to be overcome like local minima.
- **Satellite:** The FCM operator allows for better efficiency and with less failures for these five problems.
- **Summary:** As predicted by the initial experiment the problems are generally being solved more efficiently when FCM replaces `VarAddMutation`. This is confirmed by the Mann-Whitney test. The null hypothesis is rejected with a p-value of 0.00377 and a sum of ranks $W$ of 484.5. The `blocksworld-3ops` domain still has some difficult cases, possibly involving the system getting trapped in local minima.

Figure 10.6 plots median plan length of returned plans for both `VarAddMutation` and FCM on the selected problems and domains. The lines in black are the original results from Chapter 8 and the blue lines are the new results with FCM.

- **Briefcase:** GENPLAN with FCM has produced plans of much lower plan quality when compared with the original results using `VarAddMutation`.
- **Blocksworld-3ops:** The plans produced for this domain using FCM are only slightly worse than the original results using `VarAddMutation`.
- **Satellite:** Like `briefcase`, the plans are of much lower quality.
- **Summary:** All the relevant domains have experienced a decrease in plan quality as predicted by the initial results. This can be expected as FCM adds many more working actions into the population during a run. This is confirmed by the Mann Whitney test as the null hypothesis is rejected with a p-value of 0.0047 and a sum of ranks $W$ of 117.

The further results using FCM are encouraging. The efficiency of GENPLAN for all four domains has improved and has been confirmed by experiment. The downside of FCM is a loss of plan quality. These experiments are only initial results with this type of operator, and there are potentially many more that can be based on simple or even heuristic search.

### 10.2.3 Results using both FCM and Strategy 1

Both FCM and Strategy 1 individually have resulted in improved efficiency. The following experiment will try to answer the following question "Do FCM and Strategy 1 used together improve the efficiency of planning and decrease the quality of returned plans?" The purpose of the experiment is to reveal any incompatibility issues between the two improvements. The $P_r$ parameter was again set to 0.6, but a more aggressive mutation setting was used: 0.15% and 0-20 actions.

The manipulation experiment is run identically to the previous manipulation experiments. The same problems used for the previous seeding and FCM experiments are re-used here. For each problem GENPLAN was run 100 times for both settings. The experiments use the standard parameters except when it is necessary to swap the mutation component and seeding component.

The manipulation experiment will incorporate a Mann-Whitney test as set out in Chapter 6. For both experiments $\alpha$ is set to 0.05. The following hypotheses will be tested:

- **Null Hypothesis:** The median CPU time to solve planning problems for GENPLAN using `VarAddMutation` and Strategy 0 is less than or equal to the median CPU time to solve planning problems for GENPLAN using FCM and Strategy 1.
  **Alternative Hypothesis:** The median CPU time to solve planning problems for GENPLAN using `VarAddMutation` and Strategy 0 is greater than the median CPU time to solve planning problems for GENPLAN using FCM and Strategy 1.

- **Null Hypothesis:** The median plan length of resulting plans found with GENPLAN using `VarAddMutation` and Strategy 0 is greater than or equal to the median plan length of results plans found with GENPLAN using FCM and Strategy 1.
  **Alternative Hypothesis:** The median plan length of resulting plans found with GENPLAN using `VarAddMutation` and Strategy 0 is less than the median plan length of results plans found with GENPLAN using FCM and Strategy 1.

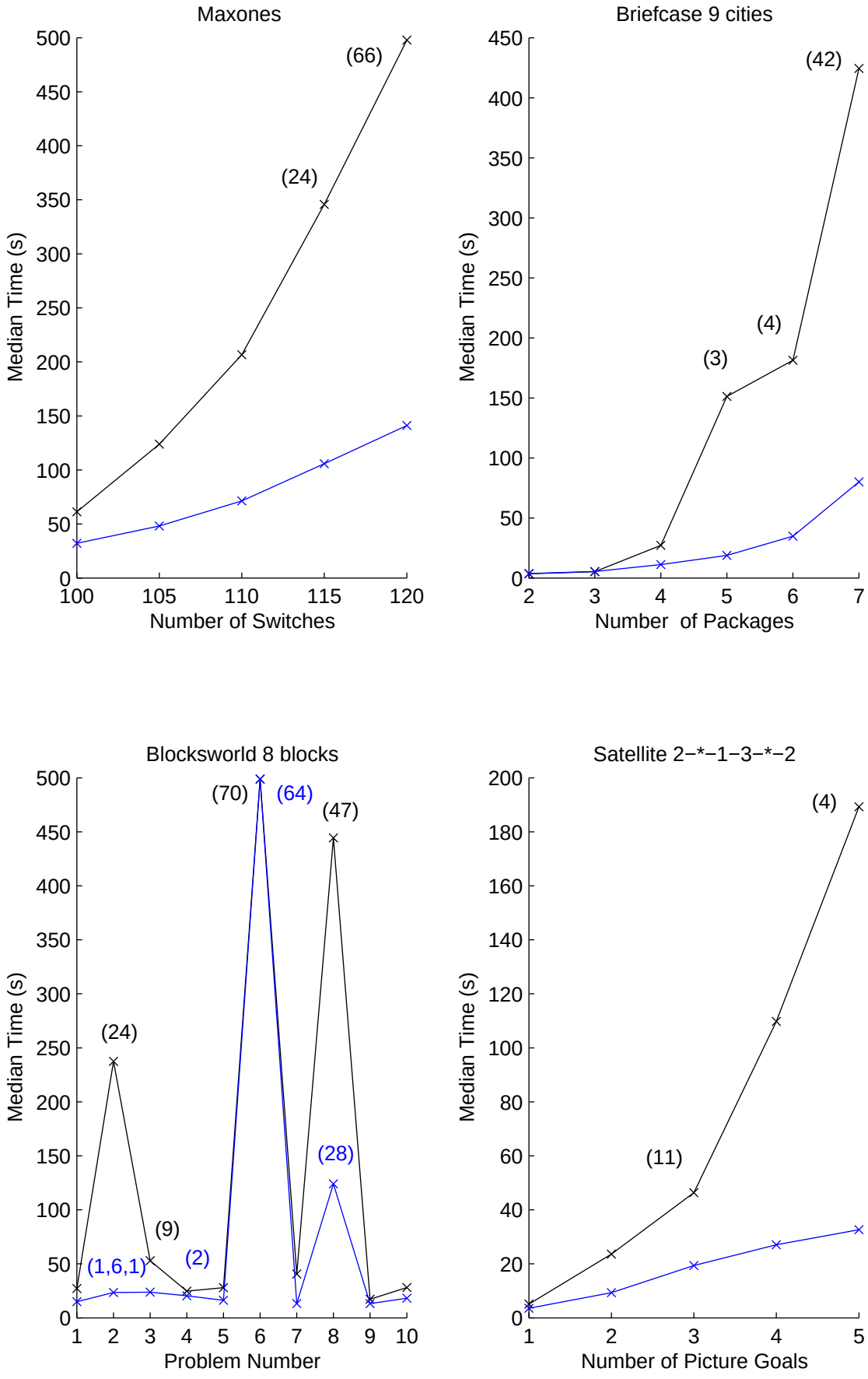



Figure 10.5: Comparing the efficiency of GENPLAN with and without FCM

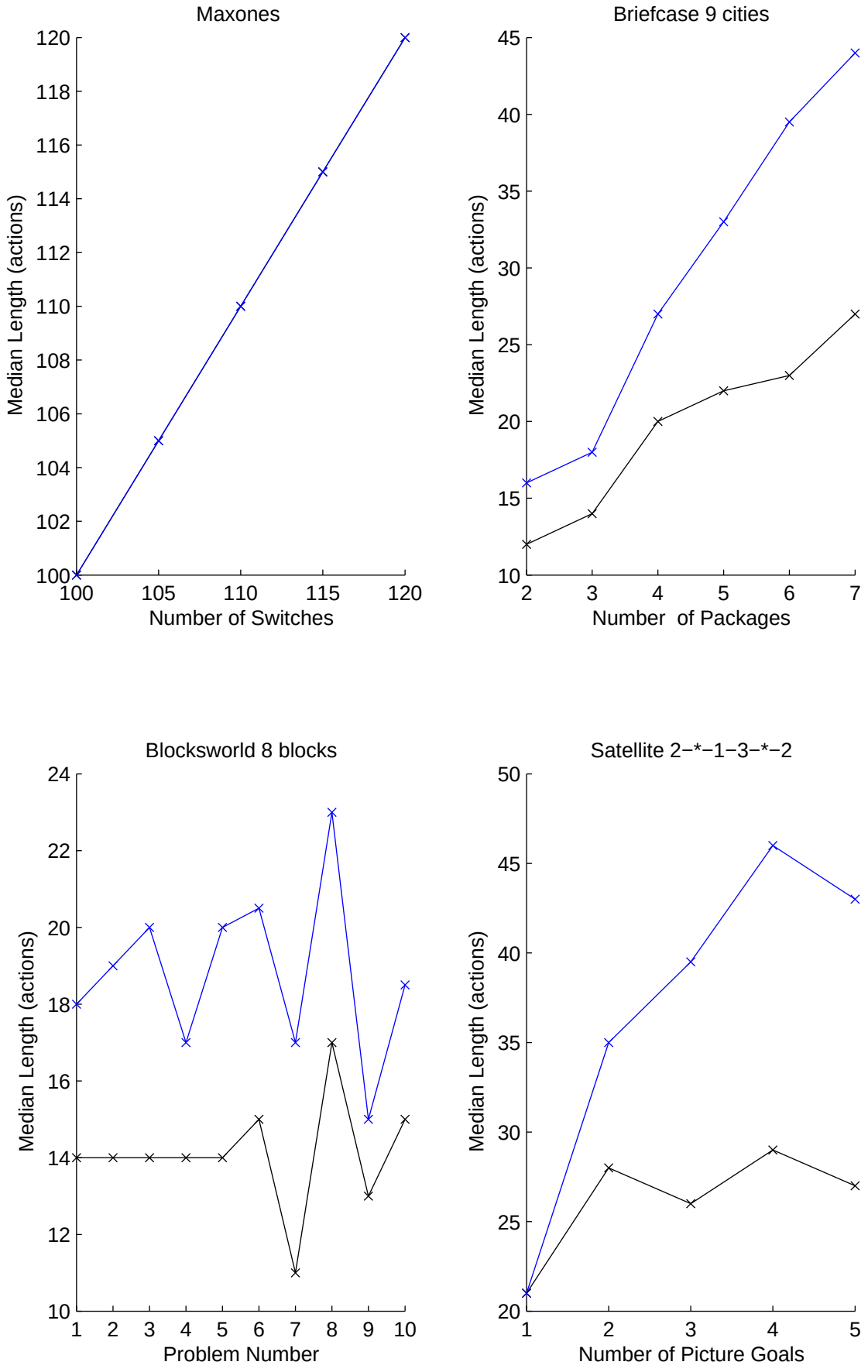



Figure 10.6: Comparing the quality of returned plans with and without FCM

As with the previous experiment. A sample of median times and a sample of median lengths will be generated using both FCM mutation and Strategy 1 and the previous results using `VarAddMutation` and Strategy 0 from Chapter 8 will make up the second sample. The sample size for the manipulation experiment involving efficiency is 26 as that is the total number of problems used. The sample size for plan quality is 21 as the results from `maxones` are not used.

Figure 10.7 plots median CPU time, with both Strategy 1 and FCM being used in place of Strategy 0 and `VarAddMutation`. The lines in black are the original results from Chapter 8 and the blue lines are the new results with Strategy 1 and FCM. The discussion refers back to Figures 10.3 and 10.5

- **Maxones:** The efficiency results when both improvements are used appear better than when using either Strategy 1 or FCM alone.
- **Briefcase:** The efficiency results when both improvements are used is almost directly equal to the Strategy 1 result and is slightly more efficient when only FCM is used.
- **Blocksworld-3ops:** The efficiency results when both improvements are used appear better than when either improvement is used on its own for this domain. Though some of the hard problems, those with many failures, are still hard.
- **Satellite:** The efficiency results when both improvements are used is almost directly equal to the seeding result and slightly more efficient when only FCM is used.
- **Summary:** In terms of efficiency, both improvements appear compatible when it comes to improving the efficiency of GENPLAN. This is confirmed by the Mann-Whitney test as the null hypothesis is rejected with a p-value $<0.0001$ and a sum of ranks $W$ of 539.

Figure 10.8 plots median plan, with both Strategy 1 and FCM being used in place of Strategy 0 and `VarAddMutation`. The lines in black are the original results from Chapter 8 and the blue lines are the new results with Strategy 1 and FCM. The discussion refers back to Figures 10.3 and 10.5

- **Briefcase:** Using both improvements has resulted in plan lengths which are roughly triple the length of the original results. Overall using both improvements together has produced the lowest quality plans.

- **Blocksworld-3ops:** The resulting plan lengths have nearly doubled for this domain when both improvements are used together as compared with the original results.

- **Satellite:** Like the `briefcase` domain, median plan lengths have increased dramatically producing the lowest quality plans so far when both improvements are used together.

- **Summary:** Using both improvements has resulted in the lowest quality plans for all three domains. This is confirmed by the Mann-Whitney test as the null hypothesis is rejected with a p-value $<0.0001$ and a sum of ranks $W$ of 32.5.

The experiment has revealed that both improvements can be used together to improve the efficiency of GENPLAN when compared to GENPLAN using the default components. However the issues with plan quality need to be resolved.

### 10.2.4 Working Actions Fitness Function

This potential improvement involves modifying the fitness function. Altering the fitness function can have a direct effect on the landscape of the search space. As shown in Chapter 6 the fitness function can be used to describe features of the search space, thus having a direct effect on the severity of benches and local minima. However altering the fitness function to create a perceived benefit may result in unexpected complications.

One improvement to the fitness function that was used with earlier versions of GENPLAN without significant investigation was to combine a working action component in to the calculation of fitness. That is, a candidate can receive some additional fitness for containing actions that succeed when simulated. This can be implemented by counting the total number of working actions in the candidate, found during simulation, and dividing by the maximum allowed plan length. The overall fitness value is now derived from a linear combination of two parts:

$$ff_{waff} = 0.8 * \frac{\text{goals achieved}}{\text{total goals}} + 0.2 * \frac{\text{number of working actions}}{\text{maximum plan length}}$$

This improvement may alleviate the issues of distraction and benches, as it should encourage GENPLAN to search over actions that change the state rather than those that fail to work.

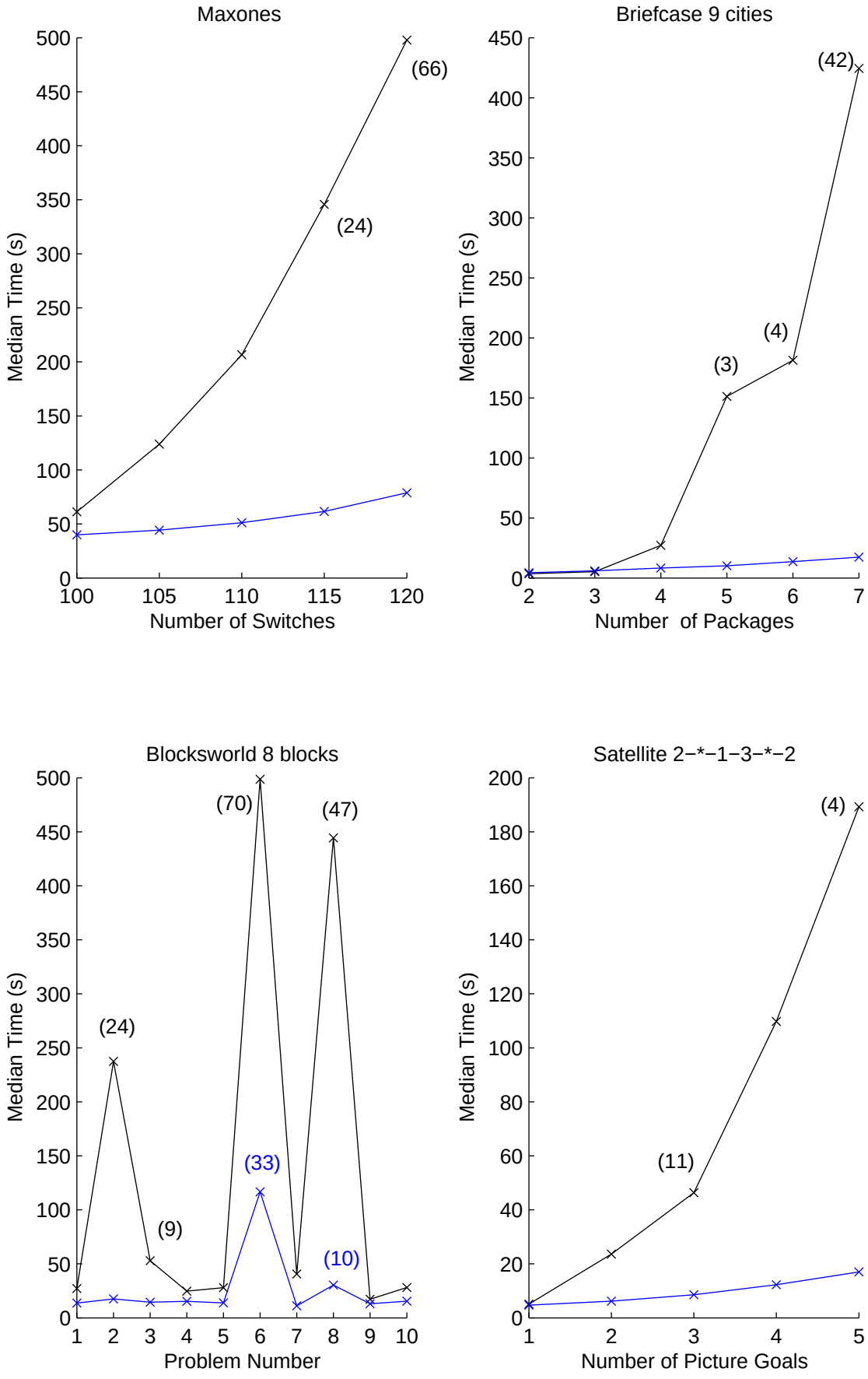



Figure 10.7: The effect on efficiency when using both FCM and Strategy 1

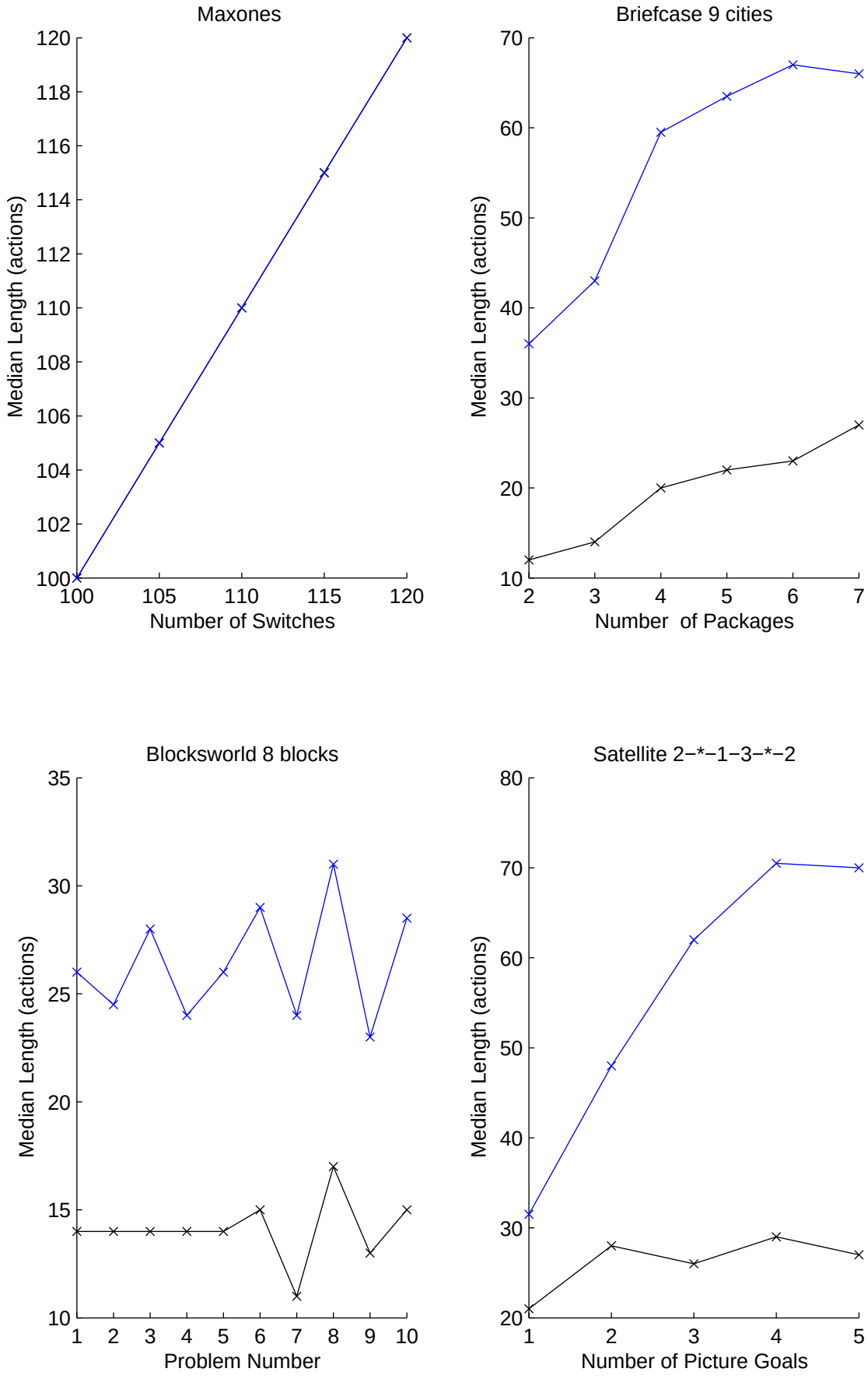



Figure 10.8: The effect on plan quality when using both FCM and Strategy 1

#### 10.2.4.1 Results

The purpose of this section is to answer the question "Does $ff_{waff}$ improve the efficiency of planning and decrease the quality of returned plans?" On the basis of previous experiments [Westerberg, 2000] it is speculated that the efficiency of GENPLAN is improved when using $ff_{waff}$. In addition, it is speculated that the quality of resulting plans decreases when using $ff_{waff}$.

The manipulation experiment compares $ff_{gc}$ with $ff_{waff}$ with all other settings at their default settings. This experiment makes use of the parameterisation problems from Chapter 7. For each problem GENPLAN was run 100 times for both settings.

The manipulation experiment will incorporate a Mann-Whitney test as set out in Chapter 6. For both experiments $\alpha$ is set to 0.05. The following hypotheses will be tested:

- **Null Hypothesis:** The median CPU time to solve planning problems for GENPLAN using $ff_{gc}$ is greater than or equal to the median CPU time to solve planning problems for GENPLAN using $ff_{waff}$.
  **Alternative Hypothesis:** The median CPU time to solve planning problems for GENPLAN using $ff_{gc}$ is less than the median CPU time to solve planning problems for GENPLAN using $ff_{waff}$.

- **Null Hypothesis:** The median plan length of resulting plans found with GENPLAN using $ff_{gc}$ is greater than or equal to the median plan length of results plans found with GENPLAN using $ff_{waff}$.
  **Alternative Hypothesis:** The median plan length of resulting plans found with GENPLAN using $ff_{gc}$ is less than the median plan length of results plans found with GENPLAN using $ff_{waff}$.

As with the previous experiment. A sample of median times and a sample of median lengths will be generated using $ff_{waff}$ and the previous results using $ff_{gc}$ will make up the second sample. The sample size for the manipulation experiment involving efficiency is 20 as that is the total number of problems used. The sample size for plan quality is 14 as the problems from `maxones` are not used. One additional results `satellite` is removed as GENPLAN did not return a result for that problem (1-10-1-3-2).

Table 10.4 presents efficiency results. The first column of results are those recorded using $ff_{gc}$. The second column results are those recorded using $ff_{waff}$.

- **Maxones:** The results using the two fitness functions are nearly identical.
- **Briefcase:** The efficiency results for $ff_{waff}$ are worse for every problem, and in some cases, introduce additional failures.
- **Blocksworld-3ops:** $ff_{waff}$ is performing slightly less efficiently than the goal counting fitness function.
- **Satellite:** The efficiency results using $ff_{waff}$ for this domain are much worse than the $ff_{gc}$ results and some problems are now unsolvable.
- **Summary:** The efficiency results appear worse for $ff_{waff}$. However this is not confirmed by the Mann-Whitney test as there is insufficient evidence to reject the null hypothesis with a p-value of 0.3102 and a sum of ranks W of 181.

Table 10.5 presents plan quality results. The first column of results are those recorded using $ff_{gc}$. The second column results are those recorded using $ff_{waff}$.

- **Briefcase:** The returned plans are of slightly worse quality when using $ff_{waff}$.
- **Blocksworld-3ops:** Again, the returned plans are of slightly worse quality when using $ff_{waff}$.
- **Satellite:** For those problems that actually return plans the plan quality is much worse.
- **Summary:** Lower plan quality results are reported for all domains when using $ff_{waff}$ in place of $ff_{gc}$. However this is not confirmed by the Mann-Whitney test as there is insufficient evidence to reject the null hypothesis with a p-value of 0.07041 and a sum of ranks W of 65.5.

The results for this improvement are not as good as first hoped. The results for `satellite` are of particular disappointment as it was hoped that this improvement would help with the bench problem. Instead the working action component appears to have had a negative effect on the efficiency of GENPLAN for this domain. The `satellite` domain is potentially a pathological case as it includes two operators `turn-on` and `turn-off` which have preconditions which are easy to satisfy. This fitness function may work better as a non-parametric measure. This means that the number of working actions is used as a tie break when choosing between two individuals with the same correctness fitness value. I now believe that any system that makes use of parametric fitness functions must be carefully examined and justified.

| Problem | Mean (failures) | std | median | Problem | Mean (failures) | std | median |
|---|---|---|---|---|---|---|---|
| 86-001 | 9.89 (0) | 2.69 | 9.58 | 86-001 | 12.85 (0) | 3.76 | 12.48 |
| 90-001 | 19.86 (0) | 5.79 | 19.7 | 90-001 | 15.94 (0) | 4.91 | 15.34 |
| 94-001 | 32.95 (0) | 10.3 | 33.25 | 94-001 | 32.78 (0) | 10.64 | 31.72 |
| 98-001 | 49.89 (0) | 18.06 | 46.22 | 98-001 | 50.85 (0) | 15.68 | 49.43 |
| 102-001 | 88.24 (0) | 36.47 | 78.93 | 102-001 | 85.25 (0) | 28.46 | 82.45 |
| 5-7-1 | 33.93 (0) | 55.86 | 22.35 | 5-7-1 | 50 (3) | 94.8 | 23.01 |
| 6-5-1 | 18.61 (0) | 5.49 | 17.68 | 6-5-1 | 21.79 (0) | 17.52 | 19.42 |
| 6-6-1 | 23.89 (0) | 22.11 | 17.7 | 6-6-1 | 25.79 (0) | 28.03 | 18.96 |
| 7-5-1 | 35.73 (0) | 32.15 | 24.31 | 7-5-1 | 49.63 (2) | 93.82 | 21.58 |
| 7-6-1 | 71.87 (0) | 74.34 | 38.88 | 7-6-1 | 149.57 (13) | 170.61 | 46.39 |
| 6-002 | 5.72 (0) | 1.71 | 5.49 | 6-002 | 6.02 (0) | 1.57 | 5.84 |
| 7-006 | 8.69 (0) | 2.77 | 8.51 | 7-006 | 9.09 (0) | 2.58 | 8.6 |
| 8-002 | 20.9 (0) | 15.4 | 16.51 | 8-002 | 18.82 (0) | 16.38 | 15.25 |
| 8-003 | 51.71 (0) | 55.84 | 25.66 | 8-003 | 67.49 (5) | 124.54 | 18.07 |
| 9-007 | 38.99 (0) | 24.51 | 30.88 | 9-007 | 46.13 (1) | 62.55 | 32.04 |
| 1-9-1-2-4 | 81.56 (0) | 44.65 | 78.61 | 1-9-1-2-4 | 493.2 (99) | 43.95 | 499.48 |
| 1-5-1-3-2 | 7.62 (0) | 5.97 | 6.09 | 1-5-1-3-2 | 95.04 (13) | 172.72 | 12.4 |
| 1-7-1-3-4 | 38.2 (0) | 23.07 | 33.66 | 1-7-1-3-4 | 461.43 (91) | 121.61 | 499.23 |
| 1-7-1-3-2 | 25.47 (0) | 17.64 | 21.09 | 1-7-1-3-2 | 428.67 (84) | 164.02 | 499.14 |
| 1-10-1-3-2 | 151.1 (1) | 104.03 | 122.98 | 1-10-1-3-2 | 497.65 (100) | 4.32 | 499.57 |

Table 10.4: CPU time results comparing $ff_{gc}$ (left column) with $ff_{waff}$ (right column)

| Problem | Mean | std | median | Problem | Mean (failures) | std | median |
|---|---|---|---|---|---|---|---|
| 86 | 86 | 0 | 86 | 86 | 86 | 0 | 86 |
| 90 | 90 | 0 | 90 | 90 | 90 | 0 | 90 |
| 94 | 94 | 0 | 94 | 94 | 94 | 0 | 94 |
| 98 | 98 | 0 | 98 | 98 | 98 | 0 | 98 |
| 102 | 102 | 0 | 102 | 102 | 102 | 0 | 102 |
| 5-7-1 | 31.31 | 3.27 | 31 | 5-7-1 | 38.22 | 7.55 | 37 |
| 6-5-1 | 28.42 | 3.79 | 28 | 6-5-1 | 32.47 | 5.16 | 32 |
| 6-6-1 | 25.53 | 3.3 | 26 | 6-6-1 | 32.02 | 8.14 | 31 |
| 7-5-1 | 31.37 | 3.53 | 31 | 7-5-1 | 39.61 | 13.56 | 37 |
| 7-6-1 | 29.21 | 3.78 | 29 | 7-6-1 | 43.34 | 17.56 | 38 |
| 6-002 | 11.87 | 3.05 | 11 | 6-002 | 13.25 | 3.5 | 13 |
| 7-006 | 13.39 | 2.42 | 13 | 7-006 | 14.08 | 2.37 | 14 |
| 8-002 | 12.05 | 2.55 | 12 | 8-002 | 14.84 | 3.41 | 15 |
| 8-003 | 13.59 | 2.53 | 13 | 8-003 | 21.35 | 20.98 | 16 |
| 9-007 | 14.42 | 2.8 | 14 | 9-007 | 18.41 | 6.73 | 17 |
| 1-9-1-2-4 | 33.72 | 10.52 | 33 | 1-9-1-2-4 | 79 | - | 79 |
| 1-5-1-3-2 | 38.74 | 12.52 | 40 | 1-5-1-3-2 | 67.66 | 25.58 | 65 |
| 1-7-1-3-4 | 41.13 | 9.67 | 41 | 1-7-1-3-4 | 83 | 31.4 | 65 |
| 1-7-1-3-2 | 38.94 | 10.34 | 40 | 1-7-1-3-2 | 85.81 | 37.33 | 76 |
| 1-10-1-3-2 | 32.1 | 11.49 | 32 | 1-10-1-3-2 | 0 | 0 | 0 |

Table 10.5: Plan quality results comparing $ff_{gc}$ (left column) with $ff_{waff}$ (right column)

### 10.2.5 Deparameterised Actions using Local Variables

Chapter 9 alluded to use of local variables in order to reduce the explicit number operands each planning operator can take thereby reducing the action space. The way in which actions are parametrised means that they are very specific and will only succeed in a limited number of states. When GENPLAN was first developed I was experimenting with `blocksworld-2ops` that allowed for local variables in the action description. The local variables are bound to objects during the simulation step. As the first version of GENPLAN did not support PDDL but rather used hard coded proceduralised versions of domains it was easy to look up variables. During the switch to PDDL it became increasingly apparent that this was a useful feature of GENPLAN. To that end I devised my own PDDL definition of local variables for PDDL. Previous work in this area had already been achieved with the first version of PDDL with `exists`, however the definition lacked precision [McDermott et al., 1998].

The `:vars` construction allows for the specification of local variables that are scoped over both the preconditions and postconditions of an action in the domain file. During the simulation step the local variable must be uniquely constrained. Local variables should not be used if the domain and problem allow for more than one binding to the local variable. In the case of `briefcase` locations are unique so it is safe to use a local variable to stand in for a location. If there were multiple briefcases and the briefcase was chosen to be the local variable then it is no longer uniquely constrained. This could be overcome if a constraint solver is combined with the simulation component. An example operator is given below using `briefcase`:

```
(:action movebriefcase
  :parameters  (?brief - briefcase  ?y - location)
  :vars(?x - location)
  :precondition (and  (at ?brief ?x))
  :effect (and  (at ?brief ?y)
                (not (at ?brief ?x))))
```

During the simulation step the local variable can be bound by comparing the preconditions with the current state. As a briefcase can only ever be at one location at one time, the location local variable can be bound to the current location of the briefcase. This location can be found in the current state.

The explicit action spaces for the two domains can be compared for the problems used in Chapter 8:

- Let $O$ be the number of objects, let $L$ be the number of locations, and let $B$ be the number of briefcases.
  - number of `putin (?o ?b ?l)` action $= O*B*L$
  - number of `takeout(?o ?b ?l)` actions $= O*B*L$
  - number of `movebriefcase(?b ?l1 ?l2)` actions $= B*L*(L-1)$
- Original Action Space: 54-198 actions
- Let $O$ be the number of objects, let $L$ be the number of locations, and let $B$ be the number of briefcases.
  - number of `putin (?o ?b)` action $= O*B$
  - number of `takeout(?o ?b)` actions $= O*B$
  - number of `movebriefcase(?b ?l)` actions $= B*L$
- New Action Space: 10-23 actions

#### 10.2.5.1 Initial Results

The first experiment involves the five problems from the parameterisation set. The purpose of which is to initially gauge the performance of the new version of the `briefcase` domain. Referring to to Table 10.6, the new version of the domain gives worse efficiency results for all five problems. More interestingly though the new version of the domain gives much tighter standard deviations suggesting more reliable behaviour. There are a couple of reasons why the efficiency results are not as good: firstly the reduced parameter versions of actions are not compatible with the action cache, which means that each action must be instantiated during simulation, secondly the simulation cost is going to be higher generally as many of the actions in a candidate have much greater chance of working. Despite these two deficiencies, results in Chapter 9 showed that reducing the action space can have a benefit on efficiency.

Referring to Table 10.7, the quality results are much more troubling than the efficiency results. Due to the fact that many more actions are likely to succeed this has had a predictably bad result on the overall quality of plans. Plan lengths have more than doubled.

| Problem | Mean (failures) | std | median | Problem | Mean (failures) | std | median |
|---|---|---|---|---|---|---|---|
| 5-7-1 | 33.93 (0) | 55.86 | 22.35 | 5-7-1 | 37.71 (0) | 9.77 | 36.65 |
| 6-5-1 | 18.61 (0) | 5.49 | 17.68 | 6-5-1 | 29.35 (0) | 7.22 | 29.22 |
| 6-6-1 | 23.89 (0) | 22.11 | 17.7 | 6-6-1 | 29.52 (0) | 11.13 | 27.06 |
| 7-5-1 | 35.73 (0) | 32.15 | 24.31 | 7-5-1 | 36.72 (0) | 8.92 | 34.99 |
| 7-6-1 | 71.87 (0) | 74.34 | 38.88 | 7-6-1 | 41.83 (0) | 10.92 | 40.68 |

Table 10.6: The left hand column presents the efficiency results using the normal domain and the right hand column presents the efficiency results using the de-parameterised version of the domain.

| Problem | Mean (failures) | std | median | Problem | Mean (failures) | std | median |
|---|---|---|---|---|---|---|---|
| 5-7-1 | 31.31 (0) | 3.27 | 31 | 5-7-1 | 69.13 (0) | 13.36 | 70.5 |
| 6-5-1 | 28.42 (0) | 3.79 | 28 | 6-5-1 | 57.57 (0) | 13.25 | 57.5 |
| 6-6-1 | 25.53 (0) | 3.3 | 26 | 6-6-1 | 63.31 (0) | 14.43 | 64 |
| 7-5-1 | 31.37 (0) | 3.53 | 31 | 7-5-1 | 68.01 (0) | 11.93 | 68 |
| 7-6-1 | 29.21 (0) | 3.78 | 29 | 7-6-1 | 70.88 (0) | 13.44 | 73 |

Table 10.7: The left hand column presents the quality results using the normal domain and the right hand column presents the quality results using the de-parameterised version of the domain.

#### 10.2.5.2 Further Results

The purpose of this section is to answer the question "Can the deparameterised version of `briefcase` be used to improve the efficiency of planning but with a decrease in the quality of returned plans?" Though the initial results were not that encouraging, the improvement could still be shown to improve the efficiency of GENPLAN at the cost of plan quality. Due to the reduced action space sizes for the larger problems instances when the differences between the sizes of action spaces of the original domain and the reduced parameter version become more extreme.

The manipulation experiment is run in a similar fashion to the previous experiments. The problems used for this second experiment are the same problems as those created for the `briefcase` domain in Chapter 8.

The manipulation experiment will incorporate a Mann-Whitney test as set out in Chapter 6. For both experiments $\alpha$ is set to 0.05. The following hypotheses will be tested:

- **Null Hypothesis:** The median CPU time to solve planning problems for GENPLAN from the `briefcase` domain using the default version is less than or equal to the median CPU time to solve planning problems for GENPLAN from the `briefcase` domain using the deparmaterised version.
  **Alternative Hypothesis:** The median CPU time to solve planning problems for GENPLAN from the `briefcase` domain using the default version is greater than the median CPU time to solve planning problems for GENPLAN from the `briefcase` domain using the deparmaterised version.

- **Null Hypothesis:** The median plan length of resulting plans found with GENPLAN using the standard version `briefcase` is greater than or equal to the median plan length of results plans found with GENPLAN using deparameterised version of `briefcase`.
  **Alternative Hypothesis:** The median plan length of resulting plans found with GENPLAN using the standard version `briefcase` is less than the median plan length of results plans found with GENPLAN using deparameterised version of `briefcase`.

A sample of median times and a sample of median lengths will be generated with the deparameterised version of `briefcase` and the results from Chapter 8 will make up the second sample. The sample size for the manipulation experiment involving

efficiency is 24 as that is the total number of problems used. The sample size for plan quality is also 24.

Referring to Figure 10.9, the black line is the original result and the blue line uses the new domain. The overheads incurred by the deparameterised version of the domain have had a negative effect on efficiency for the easier problems. The hardest problems are solved more reliably and more efficiently using the deparameterised version of the domain. However, this is not supported by the Mann-Whitney test as there is insufficient evidence to reject the null hypothesis with a p-value of 0.6086 and a sum of ranks W of 275. In general the deparameterised version of the domain has not brought about an improvement in efficiency.

The plan quality results in Figure 10.10 show that returned plans have more than doubled in length for most problems using the reduced parameter version of the domain. The reduced parameter version of the domain appears to allow for plans of lower quality. This is confirmed by the Mann-Whitney as the null hypothesis is rejected with a p-value $<0.0001$ and a sum of ranks W of 25.5. The reduced parameter version of the domain will allow for many more actions to succeed thus increasing the number of working actions in each plan and decreasing the quality of returned plans.

With additional programming to GENPLAN the reduced parameterised operators could be placed into the action cache, which would dramatically reduce the amount of overhead. To put the improvement on more solid ground it would be necessary to automate the process by which local variables are found, and this would be worth exploring.

## 10.3 Chapter Summary

This chapter has presented a series of results highlighting implemented improvements to the GENPLAN system. The results have revealed designing improvements for the system are difficult, in particular any alteration made to the fitness function should be carefully investigated.

- Improved Seeding: This improvement was based around the idea of improving the quality of the initial population using simple search. The search issues it may have addressed were distraction and benches. The results for this improvement were quite encouraging.

- Forward Chaining Mutation: This improvement also uses simple search but in

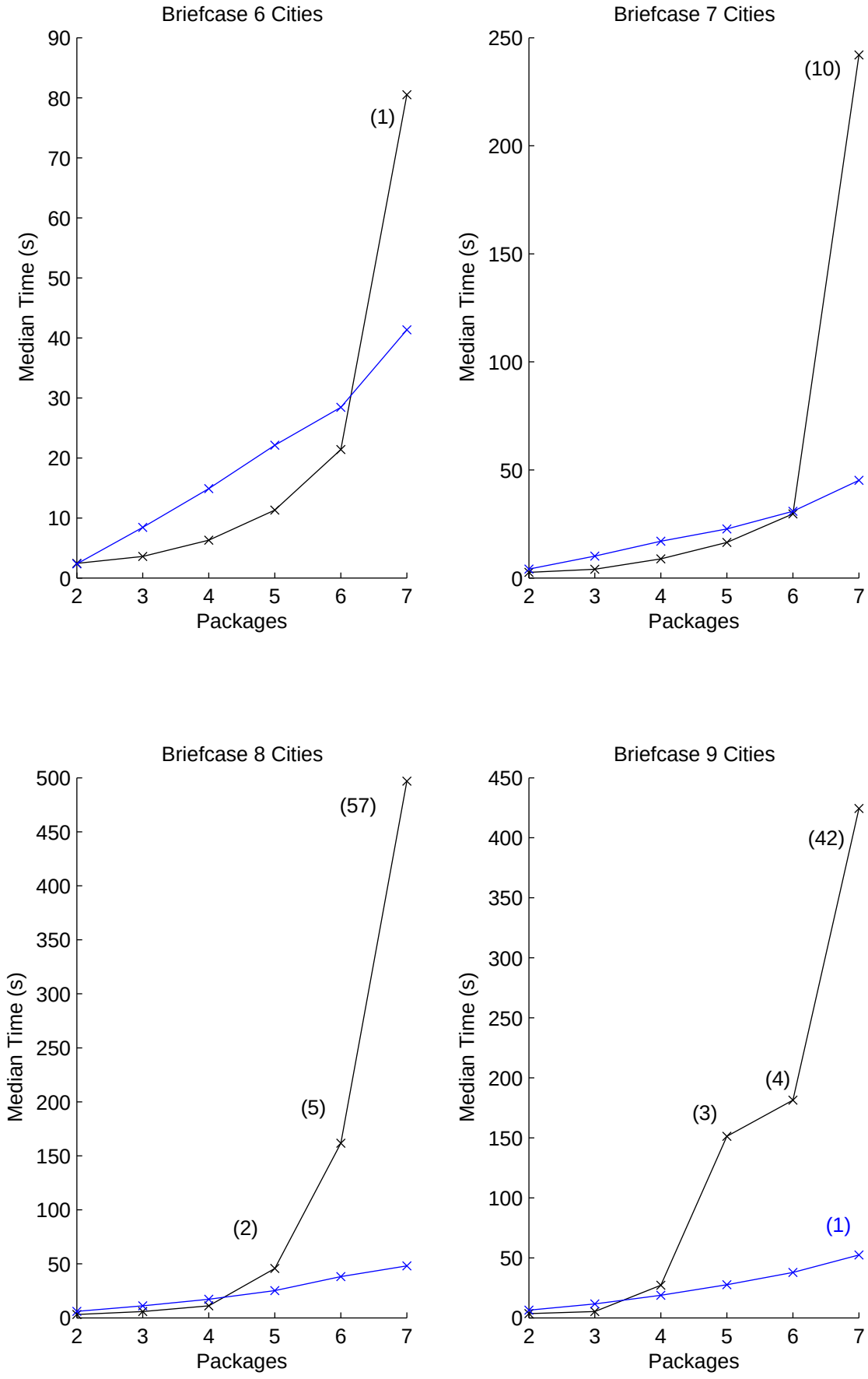



Figure 10.9: Comparing the efficiency of GENPLAN on deparameterised briefcase with normal briefcase.

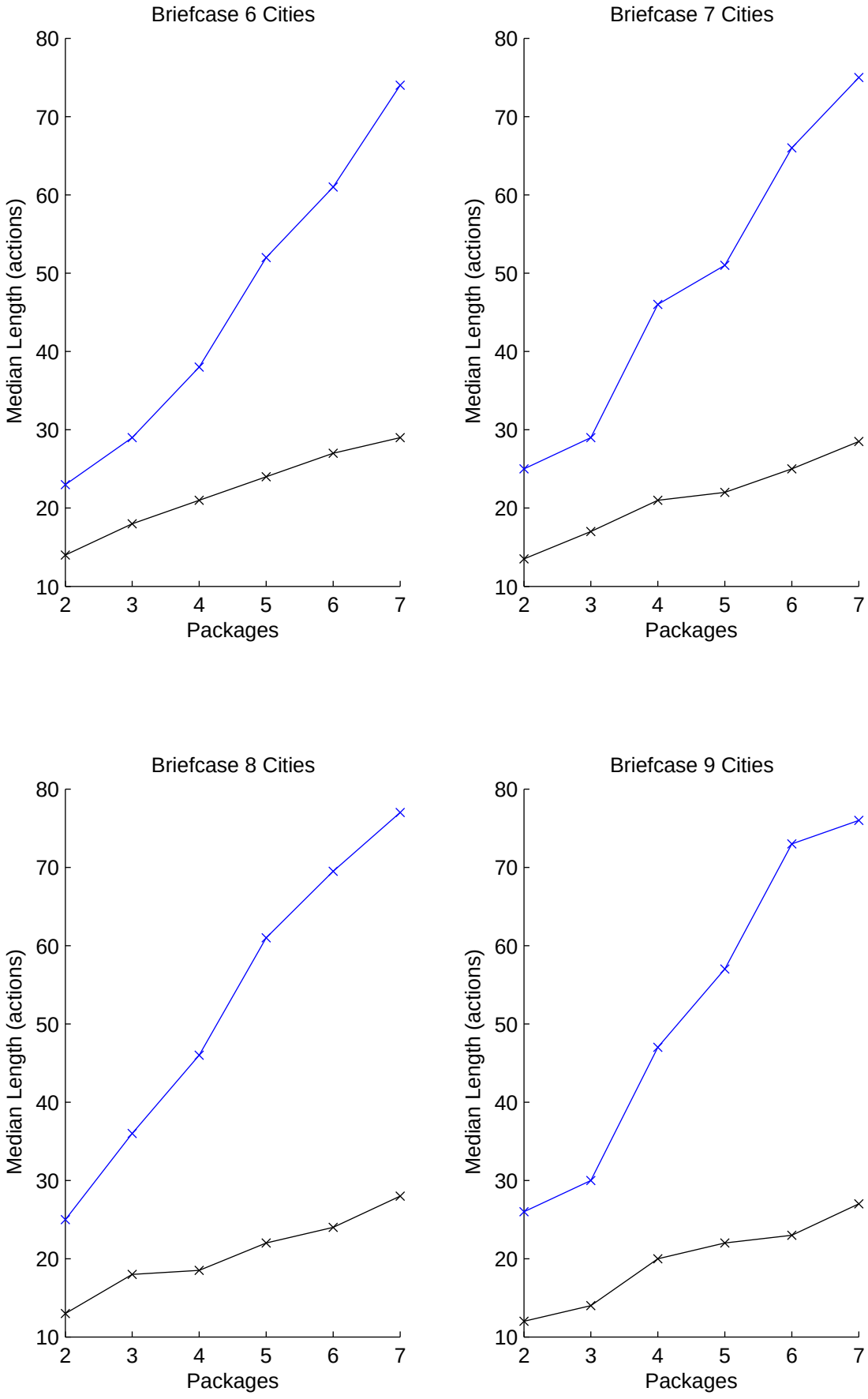



Figure 10.10: Comparing the quality of plans produced for deparameterised briefcase with normal briefcase.

order to produce a better mutation operator. This improvement could address issues involving distraction, benches, and diversity. Again, the results for this improvement were quite encouraging.

- Working Actions Fitness Function: This improvement incorporates an additional working action component into the fitness calculation. The improvement was directed towards issues of distraction and benches but produced mixed results.

- Deparameterising Domains: This improvement makes actions less explicit by incorporating local variables into the action definition. This improvement could potentially help with with problems of distraction, benches, and diversity. This improvement was not found to improve the efficiency of GENPLAN on the `briefcase` domain.

The biggest success are: improved seeding and the new mutation operator. In addition, the results in Section 10.2.3 show that improved seeding and FCM are compatible with each other. The improvement based around using local variables to reduce the number of parameters to planning operators requires further work. I would also have liked to have included further experiments with deparameterised versions of domains, but a means of devising local variables automatically should be found first. The improvements motivated the need for an improved post-processing step that improves the quality of the returned plan. The next and final chapter concludes the thesis by summarising the main contributions.

# Chapter 11

# Conclusions

The main aim of the thesis was to answer the question: "How effective is a basic EA-based planning system at solving domain independent classical planning?" It was found in the literature that EA methods have been previously applied to classical planning problems with encouraging results but the investigations lacked depth. In order to more fully answer this question it was necessary to examine the following tasks:

1. To define the classical planning problem, to define metrics which can characterise standard domains, and to summarise standard search approaches to planning.

2. To summarise the basic EA framework in terms of Genetic Algorithms and Genetic Programming.

3. To identify areas where previous work into genetic planning has been found lacking and to show the breadth of genetic planning.

4. To describe and implement an EA-based PDDL compliant testbed genetic planner called GENPLAN.

5. To present the experimental methodology used in order to experiment with the new genetic planner and to summarise the relevant statistics used to confirm the behaviour of the genetic planner.

6. To investigate and evaluate the behaviour of GENPLAN for plan synthesis. The investigation was carried out over a number of factors: different parameterisations of the algorithm, an examination of raw performance and scaling behaviour of the genetic planner, an examination of the the main search issues, and a brief examination of directed improvements.

The key classical planning areas are: the representation language used to define the planning problem, a description of domain characteristics and of standard approaches to planning. The standard planning language PDDL is supported by GENPLAN. The PDDL language makes several restrictive assumptions about the world, but is flexible enough to allow relaxations to the assumptions. Supporting the language also allows access to a large suite of planning domains. The key characteristics used to describe the domains are: complexity, local search topology (benches, local minima, dead-ends), resources, size of action space, PDDL requirement, and whether the domain had been used as benchmark previously. Using these metrics it is possible to select domains more carefully for the evaluation in relation to the heuristic used by GENPLAN. Descriptions of the standard classical planing search approaches are included to show the breadth of approaches in the planning community. Test planners are also chosen to represent some of those approaches in order to carry out the evaluation of GENPLAN.

The thesis focused on the characteristics of basic EAs, paying particular attention to Genetic Algorithms and Genetic Programming. The key differences in the way which different EAs differ from one another are: the type of representation used, the mixture of standard operators used, and whether the EA makes use of any non-standard components. The two types of representations considered were fixed length binary chromosomes for GAs and variable size trees for GP. Once the representation has been fixed the basic components of any EA-based system are: a population, initialisation, fitness function, selection, crossover, and mutation. The Schema Theorem shows the basic theoretical considerations underlying EA techniques.

## 11.1 Contributions

The introduction to this thesis presented a series of research questions that this thesis attempted to answer. The primary question being: “How effective is a basic EA-based planning system at solving domain independent classical planning?” Therefore the main contribution of this thesis is a new planning algorithm that takes its inspiration from EAs which has been investigated in terms of different parameterisations of the algorithm, an examination of raw performance and scaling behaviour of the genetic planner, an examination of the the main search issues, and a brief examination of directed improvements. Previous work in genetic planning showed a lack of rigour and investigation into the technique over a range parameterisations, domains, search issues, and specialised operators. The questions raised in the introduction are reconsidered.

1. "How well have EAs been previously applied to fully automated planning?"

   The literature survey in Chapter 4 identified that: no single system has had extensive parameterisation experiments performed with it, a single system had not been applied to a sufficient range of differing planning domains and problems, only a limited set of limitations of the approach have been presented, and that only a limited set of specialised operators have been used and sometimes with no justification. This thesis has identified these key areas and attempted to address these issues. This thesis also brings together the most comprehensive list of literature pertaining to genetic planning.

2. "How to implement an EA-based approach to fully automated planning?"

   The three most important considerations when implementing an EA-based planning system are: the representation of solutions, whether the system is for domain dependent or domain independent planning, and whether to include any non-standard operators. This question is answered by Chapters 3, 4 and 5. Chapter 3 outlined the general framework for the behaviour of EAs. Chapter 4 listed previous approaches and summarised the main defining characteristics of the different approaches. Chapter 5 describes the implementation of GENPLAN.

   This thesis presents results from a single representation, s variable length linear direct encoding, as examining multiple representations to the same level would be an enormous task. The representation used was well justified as it promised encouraging results from the outset. Whether the system will be used to perform domain dependent or domain independent planning will have a direct impact on the rest of the system, in particular on the creation of custom operators that could benefit from knowledge of the domain. Again this issue was not examined and domain independent planning was chosen as it offers the most return when examining a new approach to planning. Some specialised operators were considered in Chapter 10, but further investigation is required in this area. Any proposed improvements should be justified and examined. In the improvements area there is still much more that can be done, especially the use of different representations, and specialist operators for classical plan synthesis. GENPLAN makes use of many standard components from EAs so as not to obscure the performance of GENPLAN, and so that any further improvements can be motivated.

3. "How best to parameterise an EA to solve planning?"

A large amount of work went into exploring different parameterisations of GENPLAN in order to partly explain the behaviour of the system and to sensibly configure the system. The results of which were presented in Chapter 7. A properly configured algorithm is necessary before it can be evaluated. Parameterisation tests are time consuming and do not bring about a great gain in performance unless it is justifiable to perform the parameterisation at a per domain and per problem level. In general, I learnt that population size is an equally important parameter to set appropriately here as with other EA-based systems. A parameter of almost equal importance is the level of selection pressure which can be varied using the Tournament Size. The parameter values used in the thesis are more aggressive than those I used before the parameterisation experiments. In general the performance of GENPLAN can be improved by a more carefully selected parameterisation, but the gains do not scale well, and the system is relatively robust with most parameterisations.

4. "How to evaluate a new planning algorithm?"

   The evaluation of planners, in particular new planners, is an open issue and is still very much open to debate. Chapters 2, 6, and 8 dealt with this issue. My addition to this area was to motivate the selection of domains so that they differ from each other in a measurable way. I adopted metrics used for local search in order to characterise a great deal of domains in relation to $ff_{gc}$. The selected metrics are also an aid to understanding the behaviour of GENPLAN. I took care in generating problems for the domains as well, attempting to define typical problems for a domain and recording those decisions made. I used a mixture of randomly generated and pre-existing problem sets. Pre-existing sets can be flawed in some way, but for reasonable sets adopting those problems can be an aid to the evaluation of planners. The experimental methodology records many pragmatic decisions that need to be recorded when performing experiments. The evaluation of GENPLAN was improved through the use of statistical testing. In particular, the evaluation was carried out with the aid of a non-linearity test in order to examine the scaling behaviour of GENPLAN.

5. "How well does GENPLAN perform?"

   GENPLAN scaled best in terms of efficiency against the graphplan-based planner IPP4 and worst against the heuristic planner Metric-FF. A couple of positive results were also reported in comparison with the partial order planner, VHPOP,

and the stochastic planner BlackBox. This suggests that GENPLAN can plan but has some way to go to be competitive with the better modern planners. Generally, GENPLAN produces the worst raw quality plans of all the planners. The quality of plan produced by GENPLAN is normally a constant factor longer than that of a plan produced by a competition planner.

6. "What issues are there with the approach?"

   Chapter 9 attempted to highlight all the major search issues that must be considered when applying EAs to plan synthesis: benches, local minima, dead-ends, resources, diversity, bloat, and crossover. In this respect I regard the thesis as fairly complete. The issues of GENPLAN could be further investigated, but all the major problems I know of have been highlighted. Once these problems have been overcome the technique has to be applied to new domains involving time, and the more real-world like problems used for IPC 4 which are currently well beyond the scope of this approach.

7. "How can the problems of an EA approach be alleviated?"

   Chapter 10 presented four potential improvements to the GENPLAN algorithm. The most impressive of which was to incorporate ideas of simple search into seeding and mutation to help GENPLAN move between benches, and to focus the planner on more useful actions. Little work has been developed in order to overcome other issues like: local minima, resources, bloat, and diversity.

# Appendix A

# Domain Specification

The domain file for each domain is reproduced in this appendix. The domains are ordered in alphabetical order. Descriptions of domains were given in Chapter 6.

## A.1 Blocksworld-3ops

```
(define (domain blocksworld)
(:predicates (clear ?x)
             (on-table ?x)
             (on ?x ?y))

(:action move-b-to-b
  :parameters (?bm ?bf ?bt)
  :precondition (and (clear ?bm) (clear ?bt) (on ?bm ?bf))
  :effect (and (not (clear ?bt)) (not (on ?bm ?bf))
               (on ?bm ?bt) (clear ?bf)))

(:action move-b-to-t
  :parameters (?bm ?bf)
  :precondition (and (clear ?bm) (on ?bm ?bf))
  :effect (and (not (on ?bm ?bf))
               (on-table ?bm) (clear ?bf)))

(:action move-t-to-b
  :parameters (?bm ?bt)
  :precondition (and (clear ?bm) (clear ?bt) (on-table ?bm))
  :effect (and (not (clear ?bt)) (not (on-table ?bm))
               (on ?bm ?bt))))
```

## A.2 Briefcase

```
(define (domain typed-briefcase)
(:requirements :strips :typing)
(:types location obj - object
        briefcase item - obj)
(:predicates (at ?x - obj ?y - location)
             (in-briefcase ?x - item ?y - briefcase))

(:action putin
  :parameters (?obj - item ?brief - briefcase ?loc - location)
  :precondition (and  (at ?obj ?loc) (at ?brief ?loc))
  :effect (and (in-briefcase ?obj ?brief)
               (not (at ?obj ?loc ))))

(:action takeout
  :parameters (?obj - item ?brief - briefcase ?loc - location)
  :precondition (and (in-briefcase ?obj ?brief)
                     (at ?brief ?loc))
  :effect (and (at ?obj ?loc)
               (not (in-briefcase ?obj ?brief))))

(:action movebriefcase
  :parameters  ?brief - briefcase ?x - location ?y - location)
  :precondition (and  (at ?brief ?x))
  :effect (and (at ?brief ?y)
               (not (at ?brief ?x)))))
```

## A.3 Briefcase - Deparamaterised Actions

```
(define (domain 2p_brief)
(:requirements :strips :typing :vars)
(:types location obj - object
        briefcase item - obj)
(:predicates (at ?x - obj ?y - location)
             (in-briefcase ?x - item ?y - briefcase))

(:action putin
  :parameters  (?obj - item ?brief - briefcase)
  :vars (?loc - location)
  :precondition (and  (at ?obj ?loc) (at ?brief ?loc))
  :effect (and (in-briefcase ?obj ?brief)
               (not (at ?obj ?loc ))))

(:action takeout
  :parameters  (?obj - item ?brief - briefcase)
  :vars(?loc - location)
  :precondition (and  (in-briefcase ?obj ?brief)
                      (at ?brief ?loc))
  :effect (and (at ?obj ?loc)
               (not (in-briefcase ?obj ?brief))))

(:action movebriefcase
  :parameters  (?brief - briefcase  ?y - location)
  :vars(?x - location)
  :precondition (and  (at ?brief ?x))
  :effect (and (at ?brief ?y)
               (not (at ?brief ?x)))))
```

## A.4 DriverLog - IPC

```
(define (domain driverlog)
(:requirements :typing)
(:types location locatable - object
        driver truck obj - locatable)
(:predicates (at ?obj - locatable ?loc - location)
             (in ?obj1 - obj ?obj - truck)
             (driving ?d - driver ?v - truck)
             (link ?x ?y - location)
             (path ?x ?y - location)
             (empty ?v - truck))

(:action LOAD-TRUCK
  :parameters (?obj - obj ?truck - truck ?loc - location)
  :precondition (and (at ?truck ?loc) (at ?obj ?loc))
  :effect (and (not (at ?obj ?loc)) (in ?obj ?truck)))

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

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




(:action DISEMBARK-TRUCK
  :parameters (?driver - driver ?truck - truck ?loc - location)
```

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

(:action DRIVE-TRUCK
  :parameters (?truck - truck ?loc-from - location ?loc-to - location
               ?driver - driver)
  :precondition (and (at ?truck ?loc-from) (driving ?driver ?truck)
                     (link ?loc-from ?loc-to))
  :effect (and (not (at ?truck ?loc-from)) (at ?truck ?loc-to)))

(:action WALK
  :parameters (?driver - driver ?loc-from - location ?loc-to - location)
  :precondition (and (at ?driver ?loc-from) (path ?loc-from ?loc-to))
  :effect (and (not (at ?driver ?loc-from)) (at ?driver ?loc-to))))
```

## A.5 DriverLog - New

```
(define (domain driverlog)
(:requirements :typing)
(:types location locatable - object
        road path - location
        driver truck obj - locatable)
(:predicates (at ?obj - locatable ?loc - location)
             (in ?obj1 - obj ?obj - truck)
             (driving ?d - driver ?v - truck)
             (link ?x ?y - road)
             (path ?x ?y - location)
             (empty ?v - truck))

(:action LOAD-TRUCK
  :parameters (?obj - obj ?truck - truck ?loc - road)
  :precondition (and (at ?truck ?loc) (at ?obj ?loc))
  :effect (and (not (at ?obj ?loc)) (in ?obj ?truck)))

(:action UNLOAD-TRUCK
  :parameters (?obj - obj ?truck - truck ?loc - road)
  :precondition (and (at ?truck ?loc) (in ?obj ?truck))
  :effect (and (not (in ?obj ?truck)) (at ?obj ?loc)))

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




(:action DISEMBARK-TRUCK
```

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

(:action DRIVE-TRUCK
  :parameters (?truck - truck ?loc-from - road ?loc-to - road
               ?driver - driver)
  :precondition (and (at ?truck ?loc-from) (driving ?driver ?truck)
                     (link ?loc-from ?loc-to))
  :effect (and (not (at ?truck ?loc-from)) (at ?truck ?loc-to)))

(:action WALK
  :parameters (?driver - driver ?loc-from - location ?loc-to - location)
  :precondition (and (at ?driver ?loc-from) (path ?loc-from ?loc-to))
  :effect (and (not (at ?driver ?loc-from)) (at ?driver ?loc-to))))
```

## A.6 Gripper

```
(define (domain gripper-strips)
(:requirements :strips :typing)
(:types room ball gripper)
(:predicates (at-robby ?r - room)
             (at ?b - ball ?r - room)
             (free ?g - gripper)
             (carry ?o - ball ?g - gripper))

(:action move
  :parameters (?from - room ?to - room)
  :precondition (and  (at-robby ?from))
  :effect (and (at-robby ?to)
               (not (at-robby ?from))))

(:action pick
  :parameters (?obj - ball ?room - room ?gripper - gripper)
  :precondition (and (at ?obj ?room) (at-robby ?room) (free ?gripper))
  :effect (and (carry ?obj ?gripper)
               (not (at ?obj ?room))
               (not (free ?gripper))))

(:action drop
  :parameters (?obj - ball ?room - room ?gripper - gripper)
  :precondition (and  (carry ?obj ?gripper) (at-robby ?room))
  :effect (and (at ?obj ?room)
               (free ?gripper)
               (not (carry ?obj ?gripper)))))
```

## A.7 Maxones

```
(define (domain maxones)
(:predicates (on  ?x)
             (off ?x))

(:action switch
  :parameters (?x)
  :precondition (off ?x)
  :effect (and (on ?x) (not (off ?x)))))
```

## A.8 miconic-STRIPS

```
(define (domain miconic)
(:requirements :strips :typing)
(:types passenger floor)
(:predicates (origin ?person - passenger ?floor - floor)
             (destin ?person - passenger ?floor - floor)
             (above ?floor1 - floor  ?floor2 - floor)
             (boarded ?person - passenger)
             (not-boarded ?person - passenger)
             (served ?person - passenger)
             (not-served ?person - passenger)
             (lift-at ?floor - floor))
(:action board
  :parameters (?f - floor ?p - passenger)
  :precondition (and (lift-at ?f) (origin ?p ?f))
  :effect (boarded ?p))

(:action depart
  :parameters (?f - floor ?p - passenger)
  :precondition (and (lift-at ?f) (destin ?p ?f) (boarded ?p))
  :effect (and (not (boarded ?p)) (served ?p)))

(:action up
  :parameters (?f1 - floor ?f2 - floor)
  :precondition (and (lift-at ?f1) (above ?f1 ?f2))
  :effect (and (lift-at ?f2) (not (lift-at ?f1))))

(:action down
  :parameters (?f1 - floor ?f2 - floor)
  :precondition (and (lift-at ?f1) (above ?f2 ?f1))
  :effect (and (lift-at ?f2) (not (lift-at ?f1)))))
```

## A.9 Numeric Briefcase

```
(define (domain t_briefcase)
(:requirements :strips :typing :fluents)
(:types location obj - object
        briefcase item - obj)
(:predicates (at ?x - obj ?y - location)
             (in-briefcase ?x - item ?y - briefcase))
(:functions (distance ?x - location ?y - location)
            (travelled)
            (cost))

(:action putin
  :parameters  (?obj - item ?brief - briefcase ?loc - location)
  :precondition (and  (at ?obj ?loc) (at ?brief ?loc))
  :effect (and (in-briefcase ?obj ?brief) (not (at ?obj ?loc ))
               (increase (cost) 1)))

(:action takeout
  :parameters  (?obj - item ?brief - briefcase ?loc - location)
  :precondition (and  (in-briefcase ?obj ?brief)
                      (at ?brief ?loc))
  :effect (and (at ?obj ?loc) (not (in-briefcase ?obj ?brief))
               (increase (cost) 1)))

(:action movebriefcase
  :parameters  (?brief - briefcase ?x - location ?y - location)
  :precondition (and  (at ?brief ?x))
  :effect (and (at ?brief ?y) (not (at ?brief ?x))
               (increase (travelled) (distance ?x ?y))
               (increase (cost) 1))))
```

## A.10 Numeric Mystery

```
(define (domain n_mystery)
(:requirements :strips :typing :fluents)
(:types location obj - object
        vehicle cargo - obj)
(:predicates (at ?x - obj ?l - location)
             (conn ?l1 - location ?l2 - location)
             (in ?c - cargo ?v - vehicle))
(:functions (fuel-level ?l - location)
            (space-level ?v - vehicle)
    (space-maximum ?v - vehicle))

(:action move
  :parameters (?v - vehicle ?l1 ?l2 - location)
  :precondition (and (at ?v ?l1) (conn ?l1 ?l2)
                     (> (fuel-level ?l1) 0))
  :effect (and (not (at ?v ?l1)) (at ?v ?l2)
               (decrease (fuel-level ?l1) 1)))

(:action load
  :parameters (?c - cargo ?v - vehicle ?l - location)
  :precondition (and (at ?c ?l) (at ?v ?l)
                     (<= (+ (space-level ?v) 1) (space-maximum ?v)))
  :effect (and (not (at ?c ?l)) (in ?c ?v)
               (increase (space-level ?v) 1)))

(:action unload
  :parameters (?c - cargo ?v - vehicle ?l - location)
  :precondition (and (in ?c ?v) (at ?v ?l))
  :effect (and (not (in ?c ?v)) (at ?c ?l)
               (decrease (space-level ?v) 1))))
```

## A.11 Satellite

```
(define (domain satellite)
(:requirements :strips :equality :typing)
(:types satellite direction instrument mode)
(:predicates (on_board ?i - instrument ?s - satellite)
             (supports ?i - instrument ?m - mode)
             (pointing ?s - satellite ?d - direction)
             (power_avail ?s - satellite)
             (power_on ?i - instrument)
     (calibrated ?i - instrument)
             (have_image ?d - direction ?m - mode)
             (calibration_target ?i - instrument ?d - direction))

(:action turn_to
  :parameters (?s - satellite ?d_new - direction ?d_prev - direction)
  :precondition (and (pointing ?s ?d_prev)
                     (not (= ?d_new ?d_prev)))
  :effect (and (pointing ?s ?d_new)
               (not (pointing ?s ?d_prev))))

(:action switch_on
  :parameters (?i - instrument ?s - satellite)
  :precondition (and (on_board ?i ?s)
                     (power_avail ?s))
  :effect (and (power_on ?i)
               (not (calibrated ?i))
               (not (power_avail ?s))))




(:action switch_off
  :parameters (?i - instrument ?s - satellite)
  :precondition (and (on_board ?i ?s)
```

```
                    (power_on ?i))
  :effect (and (not (power_on ?i))
               (power_avail ?s)))

(:action calibrate
  :parameters (?s - satellite ?i - instrument ?d - direction)
  :precondition (and (on_board ?i ?s)
                     (calibration_target ?i ?d)
                     (pointing ?s ?d)
                     (power_on ?i))
  :effect (calibrated ?i))

(:action take_image
  :parameters (?s - satellite ?d - direction ?i - instrument ?m - mode)
  :precondition (and (calibrated ?i)
                     (on_board ?i ?s)
                     (supports ?i ?m)
                     (power_on ?i)
                     (pointing ?s ?d)
                     (power_on ?i))
  :effect (have_image ?d ?m)))
```

## A.12 TTD

```
(define (domain ttd)
(:requirements :strips :typing)
(:types city attraction)
(:predicates (visited ?x - attraction)
             (at ?x - city)
             (poi ?x - city ?y - attraction))

(:action go
  :parameters (?c1 - city ?c2 - city)
  :precondition (at ?c1)
  :effect (and (at ?c2) (not (at ?c1))))

(:action visit
  :parameters (?place - city ?thing - attraction)
  :precondition (and (poi ?place ?thing)
                     (at ?place)
                     (not (visited ?thing)))
  :effect (visited ?thing)))
```

## A.13 TTD_conn

```
(define (domain ttp_conn)
(:requirements :strips :typing)
(:types city attraction)
(:predicates (visited ?x - attraction)
             (at ?x - city)
             (poi ?x - city ?y - attraction)
             (conn ?x ?y - city))

(:action go
  :parameters (?c1 - city ?c2 - city)
  :precondition (and (at ?c1) (conn ?c1 ?c2))
  :effect (and (at ?c2) (not (at ?c1))))

(:action visit
  :parameters (?place - city ?thing - attraction)
  :precondition (and (poi ?place ?thing)
                     (at ?place)
                     (not (visited ?thing)))
  :effect (and (visited ?thing))))
```

# Bibliography

[Aler et al., 2002] Aler, R., Borrajo, D., and Isasi, P. (2002). Using genetic programming to learn and improve control knowledge. *Artifical Intelligence*, 141:29–56.

[Angeline, 1997] Angeline, P. J. (1997). Subtree crossover: Building block engine or macromutation. In *Second Annual Conference on Genetic Programming*, pages 9–17, Stanford, CA. Morgan kaufmann.

[Bacchus, 2000] Bacchus, F. (2000). Subset of PDDL for the AIPS2000 planning competition draft 1. Available at:http://www.cs.toronto.edu/aips2000/.

[Bacchus, 2001] Bacchus, F. (2001). AIPS'00 planning competition. *AI Magazine*, 22(3):47–56.

[Bäck et al., 1991] Bäck, T., Hoffmeister, F., and Schwefel, H. (1991). A survey of evolution strategies. In *Proceedings of the 4th International Conference on Genetic Algorithms*, pages 2–9, San Diego, CA. Morgan Kaufmann.

[Banzhaf et al., 1998] Banzhaf, W., Nordin, P., Keller, R. E., and Francone, F. D. (1998). *Genetic Programming – An Introduction; On the Automatic Evolution of Computer Programs and its Applications*. Morgan Kaufmann, San Francisco, CA.

[Barr et al., 1995] Barr, R. S., Golden, B. L., J. P. Kelly, M. G. C. R., and Stewart, W. R. (1995). Designing and reporting on computational experiments with heuristic methods. *Journal of Heurtics*, 1:9–32.

[Blum and Furst, 1997] Blum, A. L. and Furst, M. L. (1997). Fast planning through planning graph analysis. *Journal of Artificial Intelligence*, 90:281–300.

[Bonnet and Geffner, 1999] Bonnet, B. and Geffner, H. (1999). Planning as heuristic search: New results. In *5th European Conference on Planning*, Durham, UK. Springer-Verlag.

[Bonnet and Geffner, 2000] Bonnet, B. and Geffner, H. (2000). HSP: Heuristic search planner. *AI Magazine*, 21(2).

[Brie and Morignot, 2005] Brie, A. H. and Morignot, P. (2005). Genetic planning using variable length chromosomes. In *Proceedings of the 15th International Conference on Automated Planning and Scheduling*, Monterey, CA. AAAI Press.

[Burke et al., 2002] Burke, E., Gustafson, S., and Kendall, G. (2002). A survey and analysis of diversity measures in genetic programming. In *Proceedings of the Genetic and Evolutionary Computation Conference 2002*, New York, USA. Morgan Kaufmann.

[Bylander, 1994] Bylander, T. (1994). The computation complexity of propositional strips planning. *Artifical Intelligence*, 69(1-2):165–204.

[Clark, 2001] Clark, M. (2001). Construction domains: a generic type solved. In *Proceedings of the 20th Workshop of UK Planning and Scheduling Special Interest Group*, Milton Keynes, UK.

[Cohen, 1995] Cohen, P. R. (1995). *Empirical Methods for Artificial Intelligence*. The MIT Press, Cambridge, MA.

[Conover, 1999] Conover, W. J. (1999). *Practical Nonparametric Statistics*. John Wiley and Sons Inc, New York.

[Crawley, 2002] Crawley, M. J. (2002). *Statistical Computing: An Introduction to Data Analysis using S-Plus*. John Wiley and Sons Inc, New York.

[Currie and Tate, 1991] Currie, K. and Tate, A. (1991). O-Plan: the open planning architecture. *Artifical Intelligence*, 52:49–86. Available at: http://www.aiai.ed.ac.uk/~oplan/.

[Darwin, 1859] Darwin, C. (1859). *On the Origin of Species by Means of Natural Selection or the Preservation of Favooured Races in the Struggle for Life*. John Murray, London, UK.

[Edelkamp and Hoffman, 2003] Edelkamp, S. and Hoffman, J. (2003). PDDL2.2: The language for the classical part of the 4th international planning competition. Available at: http://www.informatik.uni-freiburg.de/~hoffmann/ipc-4/main.html.

[Edelkamp et al., 2004] Edelkamp, S., Hoffman, J., Littman, M., and Younes, H. (2004).

[Fikes and Nilsson, 1971] Fikes, R. E. and Nilsson, N. (1971). STRIPS: A new approach to the application of theorem proving to problem solving. *Artifical Intelligence*, 2:189–208.

[Fox and Long, 1998] Fox, M. and Long, D. (1998). The automatic inference of state invariants in TIM. *Journal of Artificial Intelligence Research*, 9:367–421.

[Fox and Long, 2001a] Fox, M. and Long, D. (2001a). PDDL2.1: An extension to PDDL for expressing temporal planning domains. Available at: http://www.dur.ac.uk/d.p.long/competition.html.

[Fox and Long, 2001b] Fox, M. and Long, D. (2001b). STAN4: A hybrid planning strategy based on subproblem abstraction. *AI Magazine*, 22(3):102–111.

[Frank et al., 1997] Frank, J., Cheeseman, P., and Stutz, J. (1997). When gravity fails: Local search topology. *Artificial Intelligence Research*, 7:249–281.

[Garrido and Long, 2004] Garrido, A. and Long, D. (2004). Planning with numeric variables in multiobjective planning. In *Proceedings of the 16th Eureopean Conference on Artificial Intelligence*, Valencia, Spain. IOS Press.

[Gent et al., 1997] Gent, I. P., Grant, S. A., MaxIntyre, E., Prosser, P., Shaw, P., Smith, B. M., and Walsh, T. (1997). How not to do it. Technical Report 97.27, School of Computer Studies.

[Ghallab et al., 2004] Ghallab, M., Nau, D., and Traverso, P. (2004). *Automated Planning: Theory and Practice*. Elsevier, San Francisco, CA.

[Goldberg, 1989] Goldberg, D. E. (1989). *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison Wesley, Boston, MA.

[Goldberg, 2002] Goldberg, D. E. (2002). *The Design of Innovation, Lessons from and for Competent Genetic Algorithms*. Kluwer Academic Publishers, Boston, MA.

[Handley, 1994] Handley, S. G. (1994). The automatic generations of plans for a mobile robot via genetic programming with automatically defined functions. In Kinnear, Jr., K. E., editor, *Advances in Genetic Programming*, chapter 18, pages 391–407. MIT Press.

[Haslum, 2004] Haslum, P. (2004). TP4'04 and $hsp^*_a$. In *4th International Planning Competition Booklet, part of ICAPS'04*, Whistler, Canada.

[Haslum and Geffner, 2001] Haslum, P. and Geffner, H. (2001). Heuristic planning with time and resources. In *Proceeding of 6th European Conference on Planning*, Toledo, Spain.

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

[Hoffman, 2003] Hoffman, J. (2003). *Utilizing Problem Structure in Planning: A Local Search Approach (LNAI 2854)*. Springer-Verlag, Berlin, Germany.

[Hoffmann, 2003] 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 and Nebel, 2001] Hoffmann, J. and Nebel, B. (2001). The FF planning system: Fast plan generation through heuristic search. *Journal of Artificial Intelligence Research*, 14:253–302.

[Holland, 1975] Holland, J. H. (1975). *Adaptation in Natural and Artificial Systems*. University of Michigan Press, Ann Harbor, MI.

[Hooker, 1995] Hooker, J. N. (1995). Testing heuristics: We have it all wrong. *Journal of Heuristics*, 1:32–42.

[Howe and Dahlman, 2002] Howe, A. E. and Dahlman, E. (2002). A critical assessment of benchmark comparison in planning. *Journal of Artificial Intelligence Research*, 17:1–33.

[Jones, 1995] Jones, T. (1995). Crossover, macromutation, and population-based search. In *Proceedings of the Sixth International Conference on Genetic Algorithms*, pages 73–80, San Francisco, CA. Morgan Kaufmann.

[Kanji, 1999] Kanji, G. K. (1999). *100 Statistical Tests*. SAGE Publications Ltd, London, UK.

[Kautz and Selman, 1992] Kautz, H. and Selman, B. (1992). Planning as satisfiability. In *Proceedings of the 10th European Conference on Artificial Intelligence*, Vienna, Austria. Wiley.

[Kautz and Selman, 1996] Kautz, H. and Selman, B. (1996). Pushing the envelope: Planning, propositional logic, and stochastic search. In *Proceedings of the Thirteenth National Confernece on Artificial Intelligence*, Portland, OR. AAAI Press.

[Kautz and Selman, 1999] Kautz, H. and Selman, B. (1999). Unifying SAT-based and graph-based planning. In *Proceedings of the 16th International Joint Conference on Artificial Intelligence*, Stockholm, Sweden. Morgan Kaufmann.

[Kochenderfer, 2003] Kochenderfer, M. J. (2003). Evolving hierarchical and recursive teleo-reactive programs through genetic programming. In *Proceedings of the Sixth European Conference on Genetic Programming (EuroGP-2003)*, pages 83–92, Essex, UK. Springer Verlag.

[Koehler et al., 1997] Koehler, J., Nebel, B., Hoffman, J., and Dimopoulos, Y. (1997). Extending planning graphs to an adl subset. In *Proceedings of Fourth European Conference on Planning*, Toulouse, France. Springer.

[Korf, 1987] Korf, R. E. (1987). Planning as search. *Journal of Artificial Intelligence*, 33:65–88.

[Koza, 1992] Koza, J. R. (1992). *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. MIT Press, Cambridge, MA.

[Langdon, 1999] Langdon, W. B. (1999). Size fair and homologous tree genetic programming crossovers. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO)*, Orlando, FL. AAAI Press.

[Langdon and Nordin, 2000] Langdon, W. B. and Nordin, J. (2000). Seeding genetic programming populations. In *Proceedings of the Third European Conference on Genetic Programming*, Edinburgh, UK. Springer.

[Langdon and Poli, 2002] Langdon, W. B. and Poli, R. (2002). *Foundations of Genetic Programming*. Springer-Verlag, Berlin, Germany.

[Levine and Humphreys, 2003] Levine, J. and Humphreys, D. (2003). Learning action strategies for planning domains using genetic programming. In *Applications of Evolutionary Computing, EvoWorkshops2003: EvoBIO, EvoCOP, EvoIASP, EvoMUSART, EvoROB, EvoSTIM*, pages 688–699, University of Essex, UK. Springer-Verlag.

[Long and Fox, 1999] Long, D. and Fox, M. (1999). Efficient implementation of the plan graph in STAN. *Artificial Intelligence Research*, 10:87–115.

[Long and Fox, 2000] Long, D. and Fox, M. (2000). Automatic synthesis and use of generic types in planning. In *Proceedings of the 5th Artificial Intelligence Planning Systems*, pages 196–205, Breckenridge, CO. AAAI Press.

[Long and Fox, 2001] Long, D. and Fox, M. (2001). Mult-processor scheduling problems in planning. In *International Conference on Artificial Intelligence*, Las Vegas, Nevada.

[Long and Fox, 2003a] Long, D. and Fox, M. (2003a). The 3rd international planning competition: Results and analysis. *Journal of Artificial Intelligence Research*, 20:1–59.

[Long and Fox, 2003b] Long, D. and Fox, M. (2003b). Exploiting a graphplan framework in temporal planning. In *13th International Conference on Automated Planning and Scheduling*, Trento, Italy. AAAI Press.

[Long et al., 2000a] Long, D., Fox, M., Sebastia, L., and Coddington, A. (2000a). An examination of resources in planning. In *Proceedings of the 19th UK Planning and Scheduling Workshop*, Milton Keynes, UK.

[Long et al., 2000b] Long, D., Kautz, H., Selman, B., Bonet, B., Geffner, H., Koehler, J., Brenner, M., Hoffman, J., Rittinger, F., Anderson, C. R., Weld, D. S., Smith, D. E., and Fox, M. (2000b). The AIPS-98 planning competition. *AI Magazine*, 21(2):13–33.

[Luke and Panait, 2001] Luke, S. and Panait, L. (2001). A survey and comparison of tree generation algorithms. In *In Proceedings of the Genetic and Evolutionary Computation Conference*, Trento, Italy. Morgan Kaufmann.

[Luke and Panait, 2002] Luke, S. and Panait, L. (2002). Fighting bloat with nonparametric parsimony pressure. In *In Proceedings The Seventh International Conference on Parallel Problem Solving From Nature*, Granada, Spain. Springer.

[Mark and Workman Jr., 2005] Mark, H. and Workman Jr., J. (Sep 2005). Chemometrics in spectoscopy, linearity in calibration: How to test for non-linearity. *Spectroscopy*.

[McDermott et al., 1998] McDermott, D., Ghallab, M., Howe, A., Knoblock, C., Ram, A., Veloso, M., Weld, D., and Wilkins, D. (1998). PDDL-the planning domain definition language. Technical Report CVC TR-98-003/DCS TR-1165, Yale Center for Computational Vision and Control.

[Mendenhall and Sincich, 1995] Mendenhall, W. and Sincich, T. L. (1995). *Statistics for Engineering and the Sciences*. Prentice Hall, Englewood Cliffs, NJ.

[Michalewicz, 1999] Michalewicz, Z. (1999). *Genetic Algorithms + Data Structures = Evolution Programs*. Springer-Verlag, Berlin, Germany.

[Minton et al., 1989] Minton, S., Carbonell, J. G., Knoblock, C. A., Kuokka, D. R., Etzioni, O., and Gil, Y. (1989). Explanation-based learning: A problem solving perspective. *Artificial Intelligence*, 40:63–118.

[Mitchell, 1996] Mitchell, M. (1996). *An Introduction to Genetic Algorithms*. MIT Press, Cambridge, MA.

[Murray, 2003] Murray, L. (2003). Abstracting control information in planning domains. In *Proceedings of the 22nd Workshop of the UK Planning and Scheduling Special Interest Group*, pages 246–255, University of Strathclyde, Glasgow.

[Muscettola et al., 1998] Muscettola, N., Nayak, P. P., Pell, B., and Williams, B. C. (1998). Remote agent: to boldy go where no AI system has gone before. *Artificial Intelligence*, 103:5–47.

[Muslea, 1997a] Muslea, I. (1997a). A general-purpose AI planning system based on the genetic programming paradigm. In *Late Breaking Papers at the 1997 Genetic Programming Conference*, Stanford University, CA. Stanford Bookstore.

[Muslea, 1997b] Muslea, I. (1997b). SINERGY: A linear planner based on genetic programming. In *Fourth European Conference on Planning*, Toulouse, France. Springer-Verlag.

[Nguyen and Kambhampati, 2000] Nguyen, X. and Kambhampati, S. (2000). Extracting effective and admissible state space heuristics from the planning graph. In *Proceedings of the Seventeenth National Conference on Artificial Intelligence*, Austin, Texas. AAAI Press.

[Nguyen and Kambhampati, 2001] Nguyen, X. and Kambhampati, S. (2001). Reviving partial order planning. In *Proceedings of the Seventeenth International Joint Conference on Artificial Intelligence*, Seattle, WA. Morgan Kaufmann.

[Nilsson, 1994] Nilsson, N. J. (1994). Telo-reactive programs for agent control. *Journal of Artificial Intelligence Research*, 1:139–158.

[O'Neill and Ryan, 2001] O'Neill, M. and Ryan, C. (2001). Grammatical evolution. *IEEE Transaction on Evolutionary Computation*, 5(4):349–358.

[Penberthy and Weld, 1992] Penberthy, J. S. and Weld, D. (1992). UCPOP: A sound, complete, partial-order planner for ADL. In *Third International Conference on Knowledge Representation and Reasoning*, Cambridge, MA.

[Pétursson, 2003] Pétursson, A. L. (2003). Enhancing l2plan. MSc Dissertation, Division of Informatics, University of Edinburgh.

[Poli and Langdon, 1997] Poli, R. and Langdon, W. B. (1997). A new schema theory for genetic programming with one-point crossover and point mutation. In *Genetic Programming 1997: Proceedings of the Second Annual Conference*, pages 278–285, Stanford University, CA. Morgan Kaufmann.

[R Development Core Team, 2004] R Development Core Team (2004). *R: A language and environment for statistical computing*. R Foundation for Statistical Computing, Vienna, Austria. 3-900051-07-0.

[Roberts et al., 2001] Roberts, S. C., Howard, D., and Koza, J. R. (2001). Evolving modules in genetic programming by subtree encapsulation. In *4th European Conference on Genetic Programming*, pages 160–175. Springer.

[Ross, 1998] Ross, P. (1998). Genetic algorithms and genetic programming. Available at: http://www.inf.ed.ac.uk/teaching/courses/gagp/notes/.

[Ruscio et al., 2000] Ruscio, L. H., Levine, J., and Kingston, J. K. (2000). Applying genetic algorithms to hierarchical task network planning. In *Proceedings of the UK Planning and Scheduling Special Interest group*, Milton Keynes, UK.

[Russell and Norvig, 1995] Russell, S. and Norvig, P. (1995). *Artificial Intelligence - A Modern Approach*. Prentice-Hall, Englewood Cliffs, NJ.

[Selman et al., 1994] Selman, B., Kautz, H., and Cohen, B. (1994). Noise strategies for improving local search. In *Proceedings of the Twelfth National Conference on Artificial Intelligence*, Seattle, WA. AAAI Press.

[Selman et al., 1992] Selman, B., Levesque, H., and Mitchell, D. (1992). A new method for solving hard satisfiability problems. In *Proceedings of the Tenth National Conference of Artificial Intelligence*, San Jose, CA. AAAI Press.

[Slaney and Thiébaux, 2001] Slaney, J. and Thiébaux, S. (2001). Blocks world revisited. *Artificial Intelligence*, 125:119–153.

[Smith and Weld, 1998] Smith, D. E. and Weld, D. S. (1998). Incremental graphplan. Technical Report UW-CSE-98-09-06, Department of Computer Science and Engineering, University of Washington.

[Smith et al., 1998] Smith, S. J. J., Nau, D., and Throop, T. (June, 1998). Computer bridge: A big win for AI planning. *AI Magazine*, 19(2):83–105.

[Spears et al., 1993] Spears, W. M., DeJong, K. A., Back, T., Fogel, D. B., and deGaris, H. (1993). An overview of evolutionary computation. In *Proceedings of the Third European Conference on Machine Learning*, Vienna, Austria. Springer.

[Spector, 1994] Spector, L. (1994). Genetic programming and AI planning systems. In *Proceedings of the Twelfth National Conference on Artificial Intelligence*, pages 1329–1334, Seattle, Washington. AAAI Press/The MIT Press.

[Stephens and Waelbroeck, 1999] Stephens, C. and Waelbroeck, H. (1999). Schemata evolution and building blocks. *Evolutionary Computation*, 7(2):109–124.

[Tate, 2000] Tate, A. (2000). Intelligible AI planning - generating plans represented as a set of constraintsts. In *in Research and Development in Intelligent Systems XVII, Proceedings of ES2000, The Twentieth British Computer Society Special Group on Expert Systems International Conference on Knowledge Based Systems and Applied Artificial Intelligence*, pages 3–16, Cambridge, UK. Springer.

[Tate et al., 2004] Tate, A., Dalton, J., de Siebra, C., Aitken, S., Bradshaw, J. M., and Uszok, A. (2004). Intelligent agents for coalition search and rescue task support. In *AAAI-2004 Intelligent Systems Demonstrator, in Proceedings of the Nineteenth National Conference of the American Association of Artificial Intelligence*, pages 3–16, San Jose, California. AAAI Press.

[Veloso et al., 1995] Veloso, M., Carbonell, J., Pérez, A., Borrajo, D., Fink, E., and Blythe, J. (1995). Integrating planning and learning: The PRODIGY architecture. *Journal of Experimental and Theoretical Artificial Intelligence*, 7(1).

[Weld, 1994] Weld, D. (1994). An introduction to least commitment planning. *AI Magazine*, 15(4):27–61.

[Westerberg, 2000] Westerberg, C. H. (2000). An investigation into the use of using genetic programming to solve classical planning problems. 4th Year Dissertation, Division of Informatics, University of Edinburgh.

[Westerberg, 2002] Westerberg, C. H. (2002). Elite crossover in genetic planning. In *The Graduate Student Workshop, Seventh Annual Genetic Programming Conference (GP-2002) and the International Conference on Genetic Algorithms (ICGA-2002)*, New York, NY. Morgan Kaufmann.

[Westerberg and Levine, 2000] Westerberg, C. H. and Levine, J. (2000). Genplan: Combining genetic programming and planning. In *Proceedings of the 19th UK Planning and Scheduling Special Interest group*, Milton Keynes, UK.

[Westerberg and Levine, 2001a] Westerberg, C. H. and Levine, J. (2001a). Investigation of different seeding strategies in a genetic planner. In *Applications of Evolutionary Computing*, pages 505–514, Lake Como, Italy. Springer.

[Westerberg and Levine, 2001b] Westerberg, C. H. and Levine, J. (2001b). Optimising plans using genetic programming. In *6th European Conference on Planning*, Toledo, Spain.

[Westerberg and Levine, 2001c] Westerberg, C. H. and Levine, J. (2001c). Optimising plans using genetic programming. In *Proceedings of the 2001 UK Workshop on Computational Intelligence*, Edinburgh, UK.

[Westerberg and Levine, 2003] Westerberg, C. H. and Levine, J. (2003). Apollo 13: A challenge domain for the planning community. In *Proceedings of the Workshop on the Competition: Impact, Organisation, Evaluation, Benchmarks, part of ICAPS'03*, Trento, Italy.

[Westerberg and Levine, 2004] Westerberg, C. H. and Levine, J. (2004). Numeric briefcase domain metric optimisation using an ea. In *Proceedings of the 23rd UK Planning and Scheduling Special Interest group*, Cork, Ireland.

[Whitley, 1993] Whitley, D. (1993). A genetic algorithm tutorial. Technical Report CS-93-103, Computer Science Department, Colorado State University.

[Wineberg and Christensen, 2003] Wineberg, M. and Christensen, S. (2003). Using appropriate statistics. Tutorial Slides given at GECCO 2003.

[Younes and Simmons, 2003] Younes, H. L. S. and Simmons, R. G. (2003). VHPOP: Versatile heuristic partial order planner. *Journal of Artificial Intelligence*, 20:405–430.

[Younger, 1979] Younger, M. S. (1979). *A Handbook for Linear Regression*. Duxbury Press, Belmont, CA.

[Yu et al., 2003] Yu, H., Marinescu, D. C., Wu, A. S., and Siegel, H. J. (2003). A genetic approach to planning in heterogeneous computing environments. In *International Parallel and Distributed Processing Symposium*, Nice, France. IEEE Computer Society.

[Yu et al., 2004] Yu, H., Marinescu, D. C., Wu, A. S., and Siegel, H. J. (2004). Planning with recursive subgoals. In *Proceedings of the 8th International Conference on Knowledge-Based Intelligent Information and Engineering Systems*, Wellington, New Zealand. Springer-Verlag.